MARCH/APRIL 2006
WWW.EOSJ.COM

ENTERPRISE
Open Source
JOURNAL

Focusing on Open Source Strategies in the Enterprise

Navigating the Open Source Software Jungle

Enterprise Grid Computing

Open Source Databases

Linux & Open Source Software:

A Strategic Decision for Consumer Electronics Manufacturers

A tci PUBLICATION

Integrated ERP & CRM Solution

for small to medium enterprises worldwide

Fully integrated ERP, CRM, supply chain management and accounting systems

Fast implementation without forcing final decisions

Reliable support via worldwide partner network

Personalized local and remote user interface

Safe-fail architecture for 100% availability

Global market capabilities:
- Multi-currency
- Multi-tax
- Multi-costing
- Multi-accounting
- Multi-organization

Scalable, flexible, and customizable for future growth

World-Class ERP & CRM — The Way You Want It

The future of your business depends on staying competitive. That means constantly adapting to today's changing global marketplace. With Compiere ERP & CRM, the premier open source business application, you shape your future based on business strategy rather than software limitations.

A fully integrated solution, Compiere ERP & CRM creates a seamless 360-degree view of both your business and your customer information. It provides your business with all the tools necessary to effectively manage resources, finances, customer information, and other critical business data across the enterprise, in the format you want, and with exactly the features and functions you need today and in the future.

Compiere's unique open source architecture and development model make it possible for you to experience these big-enterprise features and flexibility on a small-business budget.

Here's how:

- The product framework was designed for customization to meet your needs while supporting configuration changes "on-the-fly" — even in production.
- Your implementation support comes from Compiere Certified Partners, ERP experts from around the world, who participate in design and development of the Compiere solution.
- Compiere and its worldwide network stand behind the product with reliable customer support.
- You pay for value-added services only, driving down your implementation and ongoing usage costs.

Jump start your project — get training directly from "the Source":

May 15–19, 2006 | Bonn, Germany
August 21–25, 2006 | Portland, Oregon

To find out more or to locate a Compiere Certified Partner, visit **www.compiere.org**.

ORACLE PARTNER Microsoft Partner IBM Business Partner UNIX Mac Java

Change everything. True independence. True partnership. True open source. www.compiere.org

"Zend Platform ensures our website is available at all times."

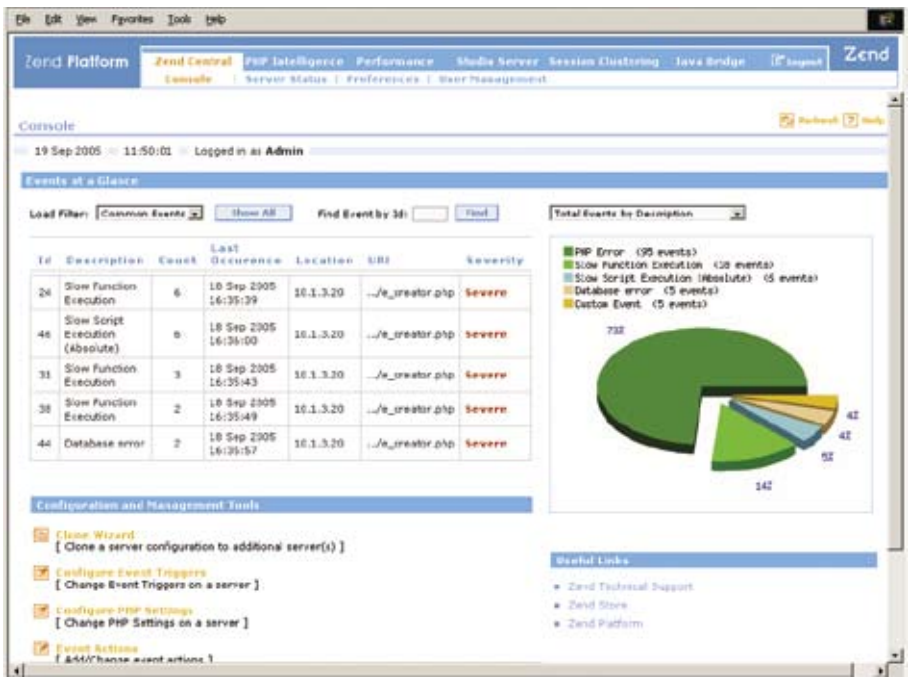
- Bret Kelly,
CTO at Marktplaats.nl, an eBay company

Zend Platform 2 - Ensuring PHP Application Availability and Response Time

Zend Platform 2 guarantees application uptime, reliability and best response time through enhanced PHP performance, monitoring, immediate problem resolution, and a unique Session Clustering scalability solution.

It uniquely removes the troubleshooting guesswork out of the equation and replaces it with peace-of-mind.

- Proactive PHP monitoring and application management
- Effective PHP Session Clustering solution
- Superior performance, scalability and reliability
- Seamless integration



Get insight into all aspects of your PHP environment

Try it now!

Download a free trial at:
www.zend.com/platform2

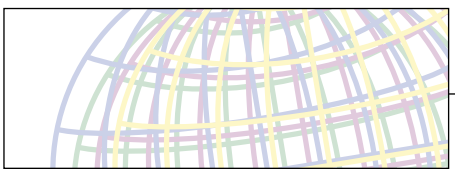
For more information, call **1-888-747-9363** or **1-408-253-8800**

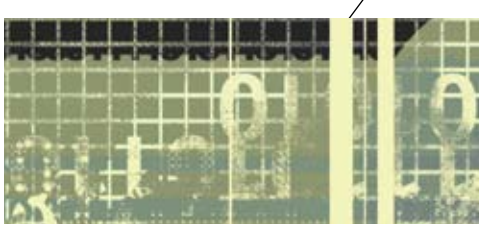
Germany: call +49-89-516199-0 Outside the US: call +972-3-613-9665

ENTERPRISE Open Source JOURNAL

MARCH/APRIL 2006
VOLUME 2 | NUMBER 2
WWW.EOSJ.COM

CONTENTS | FEATURES





I CAN'T GET MY DATA OUT OF OUR CRM SYSTEM.

CUSTOMIZING OUR CRM SYSTEM IS OFF-LIMITS.

OUR CRM SYSTEM IS SO INFLEXIBLE.

SugarCRM™ gives you control over your CRM implementation that proprietary, closed source CRM restricts. Now you can fully manage your business critical data.

Start using the most powerful commercial open source CRM solution built on the modern LAMP architecture for managing sales, marketing, support and group collaboration activities today.

Get hooked on SugarCRM. Visit: **www.sugarcrm.com/fix** or call +1 408.454.6941.

Commercial Open Source Customer Relationship Management

SUGARCRM™

www.sugarcrm.com
www.sugarforge.org

1.87SUGARCRM
+1.408.454.6941

CONTENTS | FEATURES

CONTENTS | COLUMNS


JBOSS
ORACLE
BEA
IBM
#1 in support services, according to Velociti Partners.
JBOSS
IBM
BEA
ORACLE
#1 in J2EE application server usage, according to BZ Research.
Professional Open Source from JBoss® Inc.
JBoss pioneered the Professional Open Source model that combines the cost efficiencies of open source software with the accountability and expert support services expected from enterprise software vendors. The result? Several JBoss Professional Open Source products including JBoss AS, Hibernate, and Apache Tomcat have now obtained the #1 position in their markets.
At JBoss, we understand that our success relies solely upon our ability to deliver expert support services. Whether you are trying our JEMS products or rolling them out across your enterprise, choose our industry-leading Professional Support, Consulting, and Training support services to help you every step along the way.
JBoss. Roll It Out.
To learn more, please visit our website: www.jboss.com or contact us directly at sales@jboss.com.
JBoss®

BOB THOMAS,
PUBLISHER,
EDITOR-IN-CHIEF

Publisher's Page

EOSJ Reader Survey Shows Continued Growth in Use of Open Source

As a publisher of IT trade magazines, we're always interested in learning what's happening in the world of IT. To help us gain insight into the use of open source solutions, we recently conducted a reader survey. Additionally, we asked readers about the usefulness of *EOSJ*, and what suggestions they had to help us keep the magazine useful to them. The survey was e-mailed to a randomly selected subset of *EOSJ* readers. We received 874 responses from the U.S. (56.9 percent) as well as other countries (43.1 percent).

Enterprises are without a doubt adopting and implementing many open source solutions. The most common and best known solutions, such as the Apache Web Server, MySQL, Linux, PHP and Firefox, are definitely being implemented by many companies. However, many other solutions are also being implemented such as iTracker, SugarCRM, GIMP, OpenOffice, Python, Jython, Jakarta, JBoss, ZenCart, and Thunderbird. Some enterprises are testing various open source solutions, while others are implementing some of these products into production. When we asked readers, "Do you believe your company could significantly benefit in any way from increased use of open source solutions," a resounding 85.5 percent answered "Yes."

There also seems to still be many managers across all the various departments of companies who aren't aware of open source, the various solutions available, or their benefits. When we asked, "Do you believe just about everyone in your company (corporate managers, IT managers, corporate attorneys, etc.) would benefit from learning more about using open source solutions (beyond Linux)," 76 percent answered "Yes."

The number of companies that have learned about open source, and are organized to review and implement solutions, is still greater than anyone might think. When we asked the question, "Does your company have an open source plan in place for evaluating, adopting, and licensing open source solutions," 52.9 percent said "Yes." As the number of companies create and implement a plan for evaluating, adopting, and licensing open source solutions, the quantity of solutions tested and implemented also will increase.

Readers Give *EOSJ* High Marks

After only three issues, 95 percent of the respondents said they would recommend *EOSJ* to a colleague or friend. When we asked, "How would you rate *EOSJ* on the overall quality of articles," 88 percent said the articles were "Good" or "Great." For overall breadth of article coverage, 81.1 percent said the breadth of coverage was "Good" or "Great." These are very high marks for a new magazine, and we'll continue working to achieve higher marks next year.

Where can *EOSJ* improve? Readers didn't like the Flash-based digital reader we use to deliver the digital magazine, preferring us to use a PDF. Some readers noted they aren't permitted to use Flash at their company, while others noted Flash isn't an open source product. Other readers either like or don't mind using the Flash-based reader. So, we've opted to provide both for the time being.

Thank you for your continued support, and we hope you enjoy this issue. ●

ENTERPRISE Open Source JOURNAL

Publisher
BOB THOMAS
bob@eosj.com
972.354.1024

Associate Publisher
DENNY YOST
denny@eosj.com
972.354.1030

Managing Editor
AMY B. NOVOTNY
amy@eosj.com
352.394.4478

Online Services Manager
BLAIR THOMAS
blair@eosj.com

Copy Editors
DEAN LAMPMAN
PAT WARNER

Art Director
MARTIN W. MASSINGER
martin@eosj.com

Production Manager
KYLE RICHARDSON
kyle@eosj.com

Advertising Manager
BLAIR THOMAS
blair@eosj.com
972.354.1025

Advertising Administrator
DENISE T. CULLERS

The editorial material in this magazine is accurate to the best of our knowledge. No formal testing has been performed by *Enterprise Open Source Journal* or Thomas Communications, Inc. The opinions of the authors do not necessarily represent those of *Enterprise Open Source Journal*, its publisher, editors, or staff.

Subscription Rates: Free subscriptions are available to qualified applicants worldwide at www.eosj.com.

Inquiries: All inquiries should be sent to:
Enterprise Open Source Journal
9330 LBJ Freeway, Suite 800
Dallas, Texas 75243
Voice: 214.340.2147
e-Mail: info@eosj.com

***Enterprise Open Source Journal* Article Submission Guidelines:**
Enterprise Open Source Journal accepts submission of articles on subjects related to open source. *Enterprise Open Source Journal* Writer's Guidelines are available by visiting www.eosj.com. Articles and article abstracts may be sent directly via e-mail to managing editor, Amy Novotny, at amy@eosj.com.

Paving the way for open source software innovation and self-reliance.

Geronimo
Tomcat
Spring
Hibernate
JSF/MyFaces
JBoss

Once an uncertain path pioneered by a brave few, open source software today has an established route to success in the enterprise.

Still, open source migration and integration is a journey unique to each organization. Those looking to maximize the benefits of OSS run the risk of relinquishing IT control to "semi-proprietary" companies, never truly adopting the open source way.

Virtuas enables open source self-sufficiency by empowering our clients to rely on their people and never locking IT into a single "solution".

VIRTUAS

www.virtuas.com/TakeBackIT

BY MICHAEL GOULDE

When Proprietary Goes Open

Prior to Sun Microsystems announcing it would make much of its software assets available under open source licenses, there had been only a handful of instances where companies had taken commercially licensed software and made that same software available under open source license. This was a big risk for Sun, but I think it's going to pay off in a big way in the future for the company. The argument against such a move is pretty obvious—how would you replace the revenue stream from software licenses? Customers are already paying for maintenance, so there aren't any incremental revenue opportunities from services. What's the argument in favor of doing this?

Making this move has to be based on a solid understanding of the many ways the open source model can transform a product's position in the market and impact a product's roadmap. Naturally, the two main benefits are reduced cost and increased revenue. The benefits primarily come from gaining access to additional developer resources. These developers can contribute more complete testing, faster production of patches and security fixes, enhanced functionality, and the work necessary to improve product integration.

Leveraging Developer Resources

Developer participation can be attracted from many sources. Developers working for Independent Software Vendors (ISVs) that use Open Source Software (OSS) as the foundation for their business applications have an interest in improving and enhancing the infrastructure they use. ISVs that make products that complement the open source products have an interest in improving integration, optimizing performance, and enhancing functionality. System integrators and corporate application developers have a similar motivation and probably represent a much larger number of potential community participants.

These developers represent a tremendous potential resource for the owner of the OSS. They can, and have, reduced the size of the development staff necessary to maintain and evolve the product. This lowers the cost of goods for the version that's sold with support subscriptions. Sales and marketing costs are reduced as a result of potential customers freely downloading OSS to evaluate and even pilot it. Lower cost, combined with enhanced functionality and higher quality, can increase the potential market for OSS, thereby increasing the market for support and consulting services.

Opening the Door to Innovation

Software companies have to guide their development plans with roadmaps for future releases. Difficult decisions have to be made, trading off features or functionality to achieve specific strategic objectives. Often, the options that aren't chosen may represent high risk—high payoff bets that conservative management chooses not to make. An advantage of having that software available in open source is that someone else can take it and innovate. Those innovations may or may not reappear in the original version, depending on the license used, but the market benefits, as new and innovative features or even new business models become available.

The risk that's often cited for innovating on an established base is that a second, incompatible version will complicate maintenance and support for customers. This is true only if one thinks of the innovation as a competing version of the original OSS. There's every reason for the creator of the innovative derivative to assume the responsibility for maintaining their forked version as a new and separate product. Enhancements to the original software remain accessible to the developers of the new product as open source and can easily be incorporated. But even that isn't necessary. The innovators can choose to pursue an independent path of development and handle their own maintenance. This kind of flexibility and choice are hallmarks of the open source model.

Implications for the Enterprise

Enterprises should be alert to the additional proprietary products appearing as open source and to take advantage of the opportunities this presents. A formerly closed product could become the foundation for new applications that reach production far more quickly and provide significant competitive advantage.

Newly open sourced software could become the foundation for entirely new businesses and business models. The trend toward buying and selling software as services rather than as products may be largely driven by open source editions of products that are closed today.

This approach won't necessarily be limited to commercially sold software. Business applications your company has invested in for internal use could be made available as open source and you could reap all the same benefits as a commercial software supplier. Imagine having developers from across your industry contributing to the evolution of a package that you struggle to maintain. We've started to see this in some infrastructure applications and I expect to see this happening more and more in the future with business applications.

Michael Goulde is a senior analyst with Forrester Research.
e-Mail: mgoulde@forrester.com
Website: www.forrester.com

CRM TOTAL COST OF OWNERSHIP

Comparing Open Source Solutions to Proprietary Solutions

By Bernard Golden

Today's competitive business environment requires companies to stretch their capabilities like never before. The frenetic pace of today's global economy encourages innovation and advanced resource utilization as new entrants enter a market >

WHILE SOME CRM SOLUTIONS CAN INITIALLY APPEAR SIGNIFICANTLY LESS EXPENSIVE, ADDITIONAL CHARGES FOR INTEGRATION AND OTHER ADD-ONS CAN BE CONSIDERABLE.

or existing competitors launch newer, more compelling offerings.

For more than a decade, companies have used Customer Relationship Management (CRM) software to help meet these challenges. CRM software provides companies with a complete view of their customers by supplying a centralized repository of customer interaction history where every customer interaction has an ongoing context. This ensures consistent customer communication, which can increase customer satisfaction. CRM systems also can provide early warning signs of competitive advances.

First-generation CRM systems weren't completely satisfactory. They were expensive to purchase and even more expensive to customize and administer, putting them financially out of reach for all but the largest organizations (see Figure 1).

More recently, many vendors have begun to offer CRM as a service (a hosted application). That approach requires little upfront capital investment from companies and typically looks appealing because of the quick launch time. Because they offer a low initial price point and fast uptime, these providers legitimately claim a Total Cost of Ownership (TCO) advantage compared with first-generation CRM software.

However, an emphasis on price fails to account for the variety of other factors that influence TCO. This article examines CRM TCO from a holistic perspective—one that considers system flexibility, control, and price.

TCO Factor #1: Deployment Flexibility

In today's competitive, dynamic environment, corporate computing systems must easily adapt to rapidly changing business requirements. While first-generation CRM was adaptable, making changes with client/server applications was expensive and time-consuming. Second-generation CRM vendors, who offer standardized Web browser-accessible services, purposefully restrict adaptive capabilities to sustain a standardized delivery platform to maintain low costs. The third-generation CRM Web-based solutions have low setup costs due to the versatility of the open source architecture stack, and flexibility for deploying in many different environments.

Evaluating Deployment Costs

Deployment flexibility costs should be evaluated based on business needs and the ability for a CRM deployment to constantly change to meet those needs. Evaluate the CRM solution's cost and capability for these considerations:

- **Deployment timeline:** How much upfront testing can be done before making a purchasing decision?
- **Total cost:** What will the platform, database, and services layer cost? Are there additional costs for end-user tools? Data storage? Data access?
- **Migration path:** What's the path and the costs associated with transferring a deployment to a hosted or on-premise facility?
- **Flexibility:** What are the options and costs for attaining adequate data accessibility, application customization, and system integration?
- **Maintenance and support:** What is required and how much will it cost?

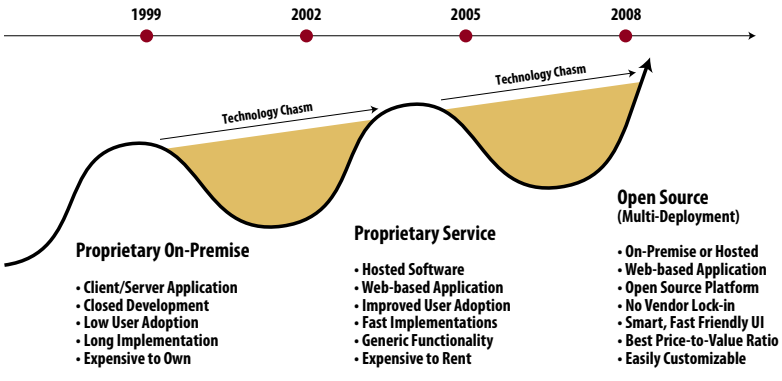


Figure 1: Popular CRM Trends

- **Restrictions:** Is there a vendor restriction on CRM enhancements or components for your deployment?

TCO Factor #2: Control

The initial deployment of a CRM product is only the first step toward realizing the full benefits. Tight integration must occur between the CRM product and existing computing systems for maximum effectiveness and payback. Integration was possible with first-generation CRM systems but these systems were closed and restrictive. Integration was consequently expensive, time-consuming and not well-suited to changing conditions.

Second-generation systems are characterized by their lack of customization flexibility. Since a customer's CRM data is stored with the vendor, integration with other systems is difficult to achieve; this also slows the pace of most integration efforts. The hosted multi-tenant environment inhibits opportunity for customization and flexibility because many customers share the same application and database (see Figure 2).

Third-generation CRM systems give organizations far greater control over the implementation options than prior CRM solutions. Source code availability and SOAP integration are key capabilities that make this possible. Each insulates an organization from vendor dependence and provides more flexibility than other offerings.

1st-generation CRM:
Closed source system
Web Services layer
Integration points
Few vendor-made integration sockets
Artificial integration costs

2nd-generation CRM:
Closed source system
Remotely hosted
Lengthy local to remote data exchanges
Few vendor-made integration sockets
Artificial integration costs

3rd-generation CRM:
Visible source code
Integrate with application, database or SOAP API
No integration socket restrictions
No additional integration costs

Figure 2: Qualities of the Three Generations of CRM Solutions

Source Code Availability

Traditionally, the act of selecting a vendor represented a shift of control from IT to the software provider. When a company commits to a particular vendor, it loses control of implementation decisions such as if and when a necessary piece of functionality is made available or when bugs are fixed.

Vendor dependence is inherent to the old mode of product delivery in which the vendor provides only a binary (the work product) and the product's source (the intellectual property) remains firmly in the vendor's control. Because the vendor retains exclusive control of product source, the customer is wholly dependent upon the vendor's decisions regarding release schedules and content.

Third-generation systems present a refreshing change to this antiquated delivery model by including source code. This frees organizations from the tyranny of vendor dependence. For instance, if a company requires an immediate bug fix, it has the ability to make the fix itself, to obtain a fix from the development community, or to wait for the fix to be included in a future release. Additionally, the company can add custom extensions even if they're not delivered in the standard product. Furthermore, customer product modifications may be provided for testing, certifying, and inclusion in the supported product.

Easy Integration

Some third-generation CRM systems bundle a SOAP interface that facilitates access by external systems. Such an interface makes integration into a company's existing computing infrastructure easier than restrictive Dynamic Link Libraries (DLLs) or Component Object Model (COM) objects. Many modern applications offer a SOAP interface and this makes integration with other applications a straightforward process.

When you compare third-generation CRM systems, always determine the completeness of the product. Look at deployment options to determine the flexibility of their security, the performance of the system, and how well it can scale to handle increased demand. Does the product provide an on-premise integration solution affording you the utmost in security and performance at no additional cost to integrate with other applications? On-demand-only solutions charge extra for this capability, and that can significantly increase the cost of your CRM deployment. Such solutions also foster vendor dependence, making it difficult to change applications integrated with the CRM system or change the CRM system itself.

TCO Factor #3: Pricing

Pricing remains a key customer concern and major component of system TCO. As noted earlier, the exceedingly high costs associated with first-generation CRM products precluded most businesses the opportunity to improve customer service. On-demand, second-generation CRM applications reduce initial capital expenditures but often result in large recurring fees. On-demand users often suffer sticker shock when the real cost of their CRM application becomes apparent in:

- An increasing number of user accounts
- A demand for data storage that exceeds the allocated amount of storage
- The need to integrate with non-standard systems.

As an organization grows, particularly by acquisitions, it can quickly outstrip the ability for first- and second-generation CRM systems to keep up with increasingly complex customization and integration requirements, more stringent security measures, and other technology initiatives. The way you'll learn this is constantly facing additional charges for customization of the product to keep pace with your needs. Furthermore, while some CRM solutions can initially appear significantly less expensive, additional charges for integration and other add-ons can be considerable. In all cases, these factors will contribute to an increased TCO. It's for this reason that all costs must be taken into consideration when evaluating a CRM solution.

Conclusion

Companies today must form and maintain strong, profitable relationships with their customers. Failing to maintain a holistic view of all customers' interactions leaves a company in a vulnerable competitive position. ●

Bernard Golden is CEO of Navica, a consulting firm offering open source strategy, implementation, and training services. He is a well-known authority on open source, particularly regarding enterprise adoption and use of open source. He is the author of *Succeeding With Open Source* (Addison-Wesley, 2005), and the forthcoming *Open Source Best Practices*.
e-Mail: bgolden@navicasoft.com
Website: www.navica.com

Global Collaborative Software Development & Deployment in the Era of Open Source: The Underwriter's Role

BY MATTHEW R. HOGG AND DANIEL EGGER

For more than 300 years, innovation in underwriting in the London market has facilitated global trade and supported new forms of economic organization. In fact, there's remarkable continuity between the first activities of Lloyd's of London and the recent development by our respective companies of the first open source insurance.

Origin of Lloyd's in Global Trade

Lloyd's of London is probably the world's best-known global insurance market, providing insurance services to businesses in more than 120 countries. Lloyd's began in the late 17th century when ship owners and their financial backers would gather to drink coffee and share the latest news in a shop owned by Edward Lloyd. British sailing ships were the cutting-edge technology of their time. Navigating the world's oceans in small wooden craft with only rudimentary navigational equipment was a capital-intensive, and high-risk—although potentially high return—business. If their "ship came in" after several years at sea, merchants could make tremendous profits, but if it were lost, they could just as easily be bankrupt.

Merchants faced an investment environment familiar to high-tech entrepreneurs today. Each voyage aimed to purchase goods at low cost in some far away port and sell them at huge profits in London, or vice versa. The London-based shipping business was highly profitable when considered as a whole, but concentration of risk made scraping together sufficient capital for any particular voyage difficult.

Merchants developed syndication as the solution. They began to purchase fractional shares in multiple voyages for the same reason modern venture capitalists back a portfolio of companies: risk diversification. Just as today's successful technology investors need specialized expertise to distinguish good deals from bad, merchants would consider the design and state of repair of each ship, the experience and judgment of its captain and crew, and the relative difficulty and potential profitability of its planned route and cargo. A contract would circulate in the Lloyd's coffee house, and those who wanted to participate would add their name to a list on the bottom of the page to "underwrite" the voyage.

Over time, individuals emerged who, for a fixed fee, would take on a portion of the risk of a ship being lost. These "underwriters" made all their money through their superior analysis of maritime risk. It's worth noting that this original insurance risk was not liability risk. No one would sue you if your ship sank; in those days no one was even necessarily at fault. Rather, it was what insurance experts would call loss-of-value, business interruption, or "first-party" risk. By assuming first-party risk, the Lloyd's underwriters made possible an exponential expansion of global trade that—along with the British Navy—made the British merchant fleet the largest in the world by tonnage—leadership it maintained for nearly 200 years.

Open Source and Global Trade

Today's comparable frontier of global trade is in the area of IT, and in particular, in the emergence of the global collaborative model for software development. What does global collaborative software development mean in practice? It means, for example, that a corporate office based in any one country can hire developers living in a second country to write software that's debugged and tested in a third country, hosted

on hardware located in a fourth country, and that's ultimately used by that corporation's employees and customers located anywhere in the world.

This model represents a profound, irreversible leveling of global boundaries to communication and commerce. And this new model relies extensively on the use of shared open source libraries, components, and applications. In fact, it'd be impossible without a shared code base made available under the license terms contained in the more than 50 open source licenses.

Because the Linux kernel and the Free Software Foundation's C Libraries and utilities are made available universally under the General Public License (GPL), developers anywhere can write applications for the combined operating system's standard Unix-based Application Program Interfaces (APIs) without fear that their projects will later become obsolete or be hijacked by any commercial entity. Open standards for interaction of software using the most critical of the Internet protocols are of practical commercial value only because the open source Apache Web Server and the Perl and Python scripting tools provide neutral, royalty-free infrastructure by which commercial developers of all kinds can use them to develop standardized, interoperable offerings.

One of the remarkable things about living in a time of fundamental transformation in the global economy is that what would have seemed extraordinary even a few years ago now seems natural and obvious. Borderless, transnational information technology based on open standards and open source licenses is easy and allows for the best resources for any given task to be brought to bear without friction or delay, and without needless duplication of effort. But the world's legal systems don't move at revolutionary speed, and are still struggling to catch up with most of the implications of this transformation.

Business decisions to invest large sums of money in new technology often depend on a prior determination that the resulting software won't need to be "open sourced" under the GPL or similar license, but instead can be maintained as proprietary intellectual property, offering its owners a source of license revenue or competitive differentiation.

Most software engineers have a good intuitive grasp of what's meant in most circumstances by a "work based on the program"—the language in the GPL that triggers a requirement that, if the new, derivative work is also "distributed," the new work must also be made available under the GPL, along with the work's underlying source code, to all those who receive it. Nevertheless, good faith differences will invariably arise, regarding either whether the proprietary code is a "work based on the program" in that specific context, or whether the company's use of the proprietary code constitutes "distribution." Good faith differences that fall within the possible scope of future judicial interpretations are often called by insurers "legal" risk—to distinguish them from risk caused by the fault of the insured.

In a global economy, it can be difficult or impossible for corporate decision-makers to anticipate whose interpretation of license language and relevant law would ultimately be binding to resolve a situation where an open source developer decides to enforce his or her perceived rights under the GPL through a lawsuit. Because open source license disputes are both global and novel, they push the world's legal systems into uncharted territory.

For example, consider the previous example, where the disputed software's original authors, current owners, hosting site, and commercial use are all in different countries. The open source developer seeking to enforce his rights may live in yet a different country. And, even if none of these activities occurred in the U.S. or Germany (the two countries that, to date, have the most experience with judicial interpretation of the GPL in compliance situations), and the developer doesn't live in either country, he might still try to bring his case there. Almost completely unresolved today are the following:

- What exactly do "work based on the program" and "distribution" mean to judges with little or no technical training?
- Does the GPL open sourcing requirement apply only to "derivative works" as that term is defined in U.S. Copyright law? (That seems to be the case in the U.S., but certain other authors' rights, such as moral rights, may come in to play in other jurisdictions.)
- How much contact must an open source developer have with the forum where he chooses to bring a lawsuit for the courts in that country to get involved?
- Is there a minimum level of contacts a defendant must have with that same jurisdiction for a court to hear the case?
- What happens if courts in more than one country claim jurisdiction over the same dispute?
- What remedies can courts impose?

Over the past year, Kiln—a managing agency at Lloyd's of London—has developed a risk-transfer product—Open Source Compliance Insurance—that can provide commercial users and developers of software anywhere in the world with global certainty regarding their risk in these uncharted areas. The goal is to let companies proceed with global collaborative software development in confidence. The Compliance Insurance coverage, like the original Lloyd's maritime coverage, is "first-party": It reimburses client companies for the loss in commercial value of proprietary software if it must be unexpectedly open-sourced.

Even with all the uncertainties surrounding legal interpretation of the GPL and other similar copyleft licenses, we've developed risk-assessment protocols that let us determine that a company's code use and ongoing development procedures put it in a zone of reasonable open source use and insurable risk. Most companies will be eligible for coverage, although the cost of coverage may vary depending on that company's current practices.

The initial target market is in mergers and acquisitions of technology companies, where representations and warranties of compliance are absolutely necessary. However, on a case-by-case basis, an offering of annual renewable coverage based on the same risk-assessment model can be made generally available. ●

Matthew R. Hogg is an underwriter at Kiln, a managing agency at Lloyd's of London. As part of the Risk Solutions team, he is Kiln's specialist in the field of Intellectual Property insurance and first-party cover.
e-Mail: matthew.hogg@kilnplc.com
Website: www.kiln.co.uk

Founder and CEO of Open Source Risk Management, **Daniel Egger** is a serial entrepreneur and lawyer and is well-versed in both the engineering and legal aspects surrounding open source use.
Voice: 919-680-4511
e-Mail: degger@osriskmanagement.com
Website: www.osriskmanagement.com

BY MARK DRIVER

What Is Open Source "Development," Anyway?

Open source licensing is distinctly different from closed source; however, other elements of the model have more in common with traditional practices and are less easily distinguished. Open source "development" is a good example. Clearly, the development practices around most successful open source efforts differ from traditional projects, yet they also share many common elements.

Open source development is a mixture of "open" principles such as open standards combined with a distributed team and a strong community that provides direct development assistance, quality control, and feature feedback. These best practices are the heart of every successful open source project.

Unlike traditional development efforts, open source projects often include a widely distributed approach with a decentralized focus. They allow overlapping efforts where the most appropriate technologies are selected downstream. This allows developers to try many approaches without having to plan ahead for all contingencies. Ultimately, the absolute rule remains true for all open source efforts: The sharing of source code is the means and not the end of the effort.

Open source is more than a license definition; it's also a methodology significantly different from the approach used by most mainstream software vendors and IT organizations. Eric Raymond's famous "Cathedral and Bazaar" paper refers to the open source phenomenon as the "bazaar" software development style and the standard commercial operating system development approach as the "cathedral" style. In the bazaar style, bugs are generally viewed as shallow phenomena because they quickly turn shallow under the scrutiny of a thousand co-developers. Thus, the maxim of the bazaar style is to release early and often. Alternatively, the typical "closed source" approach is characterized by infrequent (once or twice a year) release dates and deep, time-consuming debugging efforts after initial code construction. In contrast, the open source approach implies frequent beta releases that encourage parallel debugging efforts by the user community. Debugging can be viewed as a "parallelizable" activity that doesn't require significant coordination between debuggers.

Metcalfe's law states that the value of a network equals approximately the square of the number of users of the system ($n2 - n$). If we relate this to open source, the "users" are actually the developers within the project. As the number of developers increases, the number of "eyes" on the source code creates a *network effect* that increases the value of that source code. Some developers provide defect removal; others optimize the code for performance; while others add new features. The key is the ability to compartmentalize in modular units that minimize dependences across the project source. The degree of this modularity directly affects the potential size of the network, since complex relationships work directly against the network effect.

There are many motivators to initial open source projects. Some of the most successful open source projects began as developers "scratching their own itch." This is a core principle that binds the community together toward a common goal. This personal need is the most common factor in early project efforts. In addition, many projects begin by developers expressing their creativity. For some, the hacker ethic is as much an art form as an engineering process. Others push the ideal of free software to political and ideological extremes (e.g., The Free Software Foundation). For these developers, open source and free software represent a larger statement focused on intellectual property concerns.

Once established, an open source project focuses on building community credibility. Projects must attract like-minded developers and establish an infrastructure that provides a balance between command and control but also allows for flexibility and freedom of collaboration. Version control is the first step to allow contributors to clearly track and manage additions and changes to source. However, a larger set of design and governance structures is also required. Peer review and conflict resolution policies must be clearly set up and articulated to the community.

Based on the choices made during the inception and early organizational efforts, projects may grow rapidly, settle within a small but active niche, or die away altogether. Very few projects ever reach the critical mass of products such as Linux, Apache, or PHP. However, numerous projects enjoy successful lifecycles within small and more focused communities.

While the core "tools" of an open source development model share many common elements with traditional application development efforts, the "vision" and intent of the model has many aspects that set it apart. Traditional IT organizations can learn a great deal from these best practices. ●

Mark Driver is a research vice president with Gartner. He has more than 18 years of experience in IT focused primarily on client/server and open systems technologies. At Gartner, he covers application development tools and best practices. He also serves as the agenda manager for Gartner's open source research initiatives.
e-Mail: mark.driver@gmail.com
Website: www.gartner.com

Navigating the Open Source Software Jungle

By Ken Mulcahy

When it comes to selecting open source products, most developers and users have the same basic requirements: longevity, cost, and functionality. Ultimately, it all boils down to one basic concept: bang for the buck. The question every open source consumer must ask is, "What's the most utility I can get for the least amount of resources?"

If you're an *EOSJ* subscriber, you're probably an Open Source Software (OSS) user or developer and have been thinking about software in these terms for a while. You've probably adopted OSS as a means to satisfy many of these issues. You're likely an above-average technologist, a forward or strategic thinker, >

and not adverse to risk.

Perhaps you've already taken the OSS leap to small proof points and maybe even a reference implementation. People in your group respect your judgment and view you as a leading-edge technologist. Yet, you're still faced with selling open source solutions to the rest of the organization if you expect support and adoption.

Given the amount of Fear, Uncertainty, and Doubt (FUD) spread by your proprietary competitors and the typical lack of experience with OSS within the rest of the organization, you likely now have an uphill sales job on your hands. The management and evaluation teams will express concerns about each of these topics:

- Licenses
- Liability
- Intellectual property
- Support
- Costs (initial and recurring)
- Long-term viability
- Quality
- Reliability.

Before you address these issues, it helps to start the evaluation or analysis with an overview of what OSS actually is and the different OSS product marketing approaches you may encounter.

Convincing management that you understand various commercial market models will facilitate future discussions and ease your path to adoption. It gives everyone a common base from which to operate.

Matching OSS Solutions to Your Enterprise

The commercialization of OSS projects has generated controversy over the years. Companies have struggled to develop viable, realistic OSS commercial revenue models. Numerous models have been developed over the past five years, with several emerging as the most successful ones:

- Proprietary license model
- Subscription service model
- Services revenue model.

There are significant differences between these approaches. Each provides an infrastructure to make its implementations successful. Choosing the approach that fits your company's infrastructure, capabilities, and business model is crucial. Choosing a model that's incompatible with your needs is likely to result in failure. The rewards for implementing OSS properly are high, but so are the penalties for mistakes.

Proprietary License Model

When OSS companies market their products as completely proprietary, they generally do so to exert greater control over changes and functionality to reduce their ongoing support costs. Actually, the cost to the customer is similar to an OSS subscription service model for the same reasons.

Essentially, if you market an OSS product or solution like a proprietary product, customers will view it as a proprietary product. Customers usually aren't impressed by this type of OSS marketing and tend to view it as nothing more than a low-cost proprietary product. However, they do appreciate the huge difference in cost vs. a typical proprietary licensed product.

Only companies comfortable with relatively stable software with infrequent feature enhancements should choose this model. You should be comfortable with the product's feature set and functionality because what you see is what you get and change will occur slowly. Low cost and stability drive this solution.

In summary, with the proprietary license model, the software is proprietary and customers have no access to the source code. The software doesn't handle custom or unique environments. Proprietary support services typically include phone, e-mail, and help desk. Patches and bugs are available only in the next release. Enhancements are released to the company's roadmap. Professional services are limited and often offered through third parties, but product training is usually available.

Customers who are a good fit for this model are those that like to manage their own OSS site administration. This model provides OSS integration services with enterprise hardware and software; some OSS companies may provide integration support. With this model, the OSS provider controls application development and provides its own performance tuning.

Subscription Service Model

The subscription service model keeps the OSS software as a public project. The model relies on an initial subscription fee and a recurring annual maintenance subscription fee. While this model is similar to a proprietary software license model, it's typically significantly less expensive. In theory, this great difference in cost is due to the product company's lower expenses in the form of original engineering development, marketing, sales, and ongoing support costs, which are greatly reduced by the open source development community's efforts and those of the user base.

This model works well for a large enterprise that has a solid support infrastructure and can manage its own integration and administration, relying on help desk and revision control services for the software stack. This is similar to how commercial companies support their proprietary licensed software implementations.

Only companies that are comfortable with relatively stable software with infrequent feature enhancements should choose

When it comes to selecting open source products, most developers and users have the same basic requirements: longevity, cost, and functionality.

this model. You should be quite comfortable with the product's feature set and functionality, as they will slowly change. Subscription service model characteristics include:

- An OSS company that offers a subscription service model usually offers a supported OSS build, rather than the public project build, and sometimes includes proprietary software. Sometimes the software is extendable, depending on the OSS license type and whether the user has access to the source code, especially in the proprietary portions.
- This model doesn't support extensions or customizations because they're difficult to support and manage. Custom or unique environments aren't feasible with this model and support services are often limited in scope and similar in nature to proprietary support services.
- Customers who are a good fit for this model are those able to manage their own OSS site administration. Customers will find that OSS integration services are available with enterprise hardware and software and some OSS companies may provide integration support. Solution providers in this model control application development and provide general enterprise support. Typically, a subscription services OSS company doesn't have a large professional services component built into its business model. This lets it maintain a relatively low support cost infrastructure so it can focus on product development, maintenance, and enhancement.

Services Revenue Model

The services revenue model is truly a pure-play OSS product model; it retains all the attributes of a non-commercial project with the principal exceptions being that the products are commercially supported and are generally driving standards. A classic example are the Apache projects because they drive commercial standards and enjoy widespread adoption. This sets the stage for widely available commercial support coupled with stability and a rich feature set.

While some may think enterprise adoption takes courage, the rewards can be extremely high. The services revenue model embraces the community as a whole, bringing the Small and Medium-size Businesses (SMBs) and enterprise users together to develop, extend, and support a rich set of functionality.

The real goal for commercializing OSS is to establish a standard to which all users can work. Vibrant open source communities, not competing proprietary ones, establish standards. Standards-based OSS will consistently deliver highly useful, complex software functionality at a relatively low initial cost, coupled with low, ongoing support costs.

When the user community is large and heterogeneous, markets tend to be large and growing. Organizations demand services at many levels, with room for large and small participants. Encouraging service competitors tends to grow the market, extending the service providers' market reach.

The measurements for success for any software program and particularly for OSS programs are the adoption rate and utility in their respective user communities. Commercial service model companies that focus on customer enablement and active community participation really facilitate adoption and growth.

Companies that strive for leading-edge OSS functionality and are interested in contributing to an active community environment should choose this model. While many companies are often satisfied with an OSS product's initial features and functionality, many others are interested in enhancing and expanding that functionality to fit their needs. This scenario can provide the best of both worlds for companies wanting to use a product at low cost that will grow well into the future.

The emergence of the service model fills a huge gap in the adoption of OSS in that the commercial companies supporting this model base their entire business models on professional services and are well-positioned to deliver sophisticated commercial solutions, drive standards, and enable customers to be self-sufficient. This provides customers with solutions at a relatively low cost that are likely to be technologically current.

- This is not a "what you see is what you get" scenario, as with the other models. Features and functionality grow rapidly due to a large user and developer base. Low cost, functionality, and growth drive this solution. With the service model, software is actually open source; the customer has complete access to source code. The OSS service model company supports a complex set of professional services, including administration, implementation, integration, consulting, extension, application development, performance tuning and testing. Standard support services typically include phone, e-mail and help desk. Patches and bugs are rapidly addressed. Enhancements generally occur in keeping with an OSS release schedule and training is available.

Customers who are a good fit for this model will take responsibility for OSS site administration and actively participate in the OSS project by, for example, making feature requests, managing patches or otherwise contributing to the OSS development. With this model, the OSS provider controls application development and project integration may be a joint responsibility between the customer and OSS provider.

Conclusion

The promise of OSS is delivering highly useful, complex software functionality at a relatively low initial cost to customers, coupled with self-sufficiency and low, ongoing support cost.

The measures of success for any software program, particularly an OSS program, are the adoption rate and utility in its respective user community. For OSS programs, this is accomplished best through ongoing customer enablement programs. The goal is to facilitate program adoption, use, and self-sufficiency for the developer and user communities.

Commercial OSS products have matured considerably. Many companies are adopting and commercially supporting OSS. Choosing the right OSS strategy for your company is vital to success. A great deal of opportunity is available for companies that can take a methodical approach to adopting OSS products and solutions that solve real problems in their environments.

The business rewards in cost, competitive advantage, time to market, and performance can be outstanding if your OSS is correctly implemented. Many commercial models for OSS exist today, so proper diligence and alignment to your company's objectives are critical.

Ken Mulcahy is vice president of Sales at Virtuas Solutions, an enterprise OSS solutions company. He has held a variety of management, sales, and marketing roles, spanning a career of more than 20 years. During the past five years, he has held key leadership positions at open source firms, including president of Tungsten Graphics, an open source graphics engineering firm, and regional manager at VA Linux, one of the industry's most successful open source companies.
e-Mail: ken@virtuas.com
Website: www.virtuas.com

BY DENNY YOST

Performance Fuels World's Leading Electronic Marketplace

IntercontinentalExchange (NYSE: ICE) is the world's leading electronic marketplace for energy trading and price discovery. ICE's transparent and efficient market structure, coupled with its secure, electronic trading platform, allow ICE to provide market participants with direct access to energy futures and thousands of Over-the-Counter (OTC) commodity products for oil and refined products, natural gas, power, and emissions. Today, more than 30 percent of the world's oil trading is handled by ICE.

Today, more than 30 percent of the world's oil trading is handled by ICE. It is easy to see how performance is a key and constant issue for the company.

It is easy to see how performance is a key and constant issue for the company.

ICE requires high-performance out of its technology and exchange systems. "We are constantly looking to improve our technology and we focus on best-of-breed solutions," says Edwin Marcial, senior vice president and CTO. "We don't take anything at face value. In order to prove a technology is best-of-breed, we put it through a rigorous proof-of-concept process. Each technology must prove itself in our intended application before we adopt it into our core systems. This goes for both open source and proprietary solutions. We simply want the best products that deliver the best performance and the most reliability."

In 2002, ICE embarked on a new project that drove the IT department to try a new application server solution. "We had a couple of developers who suggested we should replace our current application server solution with an open source software application server product named JBoss. They argued that JBoss was easier to work with, more reliable, and true to the standard," says Marcial. "We were not necessarily unhappy with our current application server, and I was hesitant to move mission-critical systems to an unproven product and open source paradigm. After months of prodding by my developers, I decided to test the JBoss application server on a pilot project for a market data feed to wireless devices. This was not a mission-critical application for us, so it was a low risk way to put the JBoss application server to the test. The pilot went very well—completing very quickly and with little issues. Over time, we implemented other projects with JBoss and our confidence in the product and open source in general grew."

The success of using JBoss for an initial project led to the product being used for mission-critical applications. "Today, we use JBoss to drive our new Website, which delivers more dynamic content than in the past," Marcial continues. "The entire Website runs on JBoss Apache Web servers and JBoss Application Server. JBoss also powers our Clearing Infrastructure as well as our Market Price Validation service, which provides a method of pricing long-dated trades."

While ICE has a best-of-breed strategy for selecting solutions, the company does feel open source software gives them a competitive advantage. "Open Source solutions allow us to be more efficient and have more control over our software solutions," says Marcial. "We want products that are best-of-breed, fast, reliable, and come with excellent support—JBoss gives us that." ●

For more information, contact JBoss Inc., 3340 Peachtree Road, NE, Suite 1200, Atlanta, GA 30326. Voice: 404-467-8555; Website: www.jboss.com.

Enterprise Grid Computing: State-of-the-Art

By Rajkumar Buyya, Ph.D., and Krishna Nadiminti

The term "grid" as used today means many different things to different people. It's often used to refer to various forms of distributed systems, such as cluster-based systems, Point-to-Point (P2P) networks, wide-area distributed storage solutions, and the like. Numerous large companies add to the confusion by liberally using the term grid while describing their products and services. So, it has now become such a common industry ›

buzzword that the actual meaning needs to be inferred from the context.

Let's provide yet another definition, one that encompasses several existing definitions and describes some basic attributes of a grid:

"Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed, 'autonomous' resources dynamically at run-time depending on their availability, capability, performance, cost, and users' quality-of-service requirements." (Source: Grid Computing Info Centre: Frequently Asked Questions, accessible at www.gridcomputing.com/gridfaq.html)

Given this definition, today's distributed systems have a varying degree of grid-like characteristics. There are many systems developed and deployed for various purposes and myriad names have emerged to describe these: compute grid, data/storage grid, campus grid, enterprise grid, global grid, knowledge grid, sensor grid, cluster grid, PC grid, commodity/utility grid, and so on. Ian Foster has written an interesting article titled "What Is the Grid? A Three Point Checklist," that describes some characteristics of a grid system. (You can access this article at www.gridtoday.com/02/0722/100136.html.)

This article lists the benefits of grid computing for the enterprise and describes a grid-oriented open source project with a compelling service-oriented framework.

Grid Benefits and Challenges

In a typical Small or Medium-Size Enterprise (SME), there are many resources that are generally underutilized for long periods. A resource in this context means any entity that could be used to fulfill any user requirement; this includes compute power (CPU), data storage, applications, and services. An enterprise grid can be loosely defined as a distributed system that aims to dynamically aggregate and coordinate various resources across the enterprise and improve their utilization for greater productivity.

Grid computing technology provides enterprises an effective solution for aggregating distributed resources and prioritizing allocation of resources to different users, projects, and applications based on their Quality of Service (QoS) requirements. These benefits ultimately result in huge cost savings for the business.

There are various applications of grid computing. Many commercial compute-intensive applications, such as drug discovery, clinical modeling, simulation, investment and credit-risk analysis, large-scale document processing, and data-intensive applications that involve aggregation and management of distributed data storage centers, can vastly benefit from the performance enhancements and resource aggregation capabilities that can accrue from the use of grid technologies.

However, grids today don't address all the issues important to the enterprise. That's because they were born in academic communities where such issues aren't a high priority. There are some important distinctions between the types of grids used in research communities and those that can be used in an enterprise or commercial environment. Figure 1 outlines the characteristics that differentiate an enterprise grid from a research-oriented grid. The stars reflect the importance/desirability of the attribute to each type of grid.

	ENTERPRISE GRID SYSTEMS	NON-COMMERCIAL GRIDS
Criticality of efficient and optimal resource usage	★★★★★	★★★★★
Sharing of inter-organizational resources	★★★	★★★★★
Authentication and authorization	★★★★★	★★★
Security of stored data and programs	★★★★★	★★★
Secure communication	★★★★★	★★★
Centralized / semi-centralized control	★★★	
License Management issues	★★★★★	★★★
Auditing	★★★★★	★★★
Quality of Service (QoS) and Service Level Agreements (SLAs)	★★★★★	★★★
Economy-based & Service-Oriented Architecture (to support QoS)	★★★★★	★★★
Interoperability of different grids (and, hence, the basis of open standards)	★★★★★	★★★★★
Support for transactions	★★★★★	★

Note: A maximum of five stars means the particular feature/attribute is of utmost importance. Absence of a star means the attribute isn't required/desirable.

Figure 1: Comparison of Features/Characteristics of a Commercial and Non-Commercial Distributed Systems

The Current State of the Enterprise Grid

Grid technology is evolving to provide solutions that more fully address enterprise requirements. The technology is rapidly moving from academia and scientific research and applications toward mainstream enterprise applications with a special emphasis on Service-Oriented Architectures (SOAs) and utility computing. The enterprise grid currently includes a range of applications that use data centers and application clusters to distribute workloads of applications such as:

- Accounts receivable
- Investment portfolio risk analysis
- Pricing securities in the finance and insurance sector
- Finding solutions to bottlenecks in product design and development cycles in the manufacturing sector
- Drug discovery in the pharmaceutical sector
- Digital media creation, rendering, and distribution management.

While most early-adopters are still running batch-oriented applications, the concepts of SOA and virtual organizations are already being used to explore the possibilities of running transactional and interactive applications on enterprise grids where the QoS is expected to be reliable, especially when bound by Service Level Agreements (SLAs).

Investment in enterprise grids is expected to grow manifold in the next five years as more companies come up with value-added

services, according to a *Network World* article titled "Grid Taking Shape in Enterprise Nets" (accessible at www.networkworld.com/news/2005/101005-grid.html). Various major companies are already offering a range of services such as:

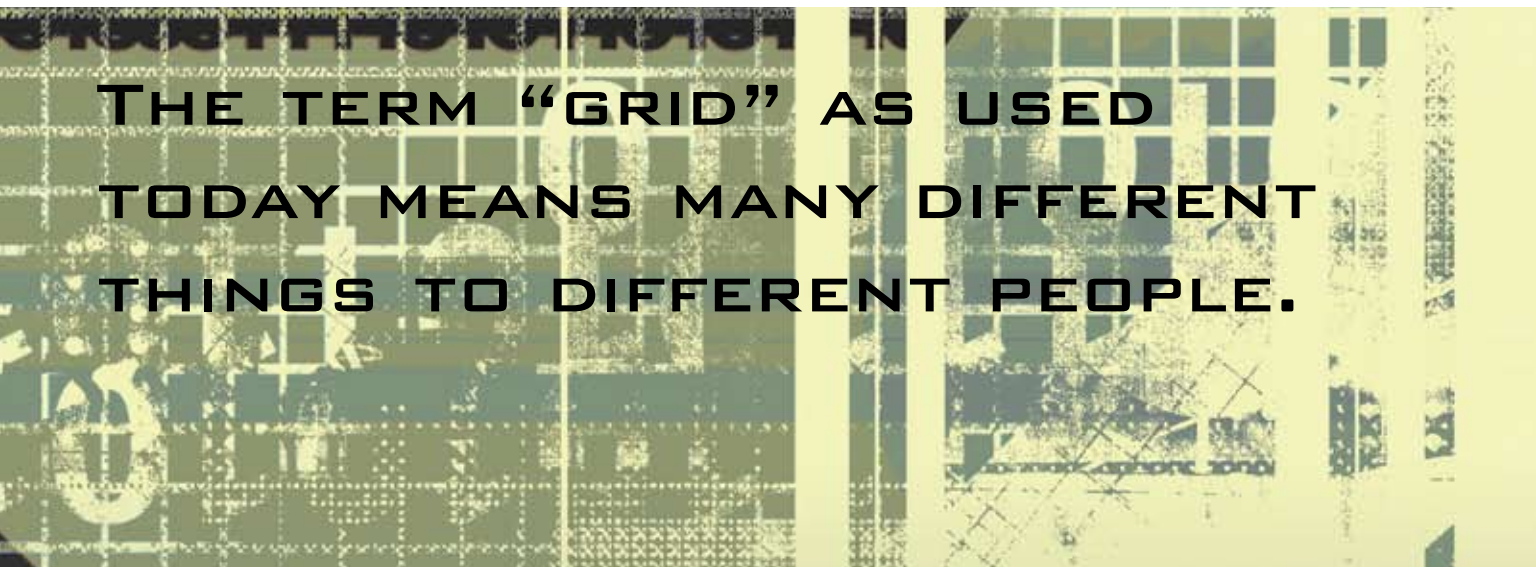
- IBM's "Grid and Grow," (described at www-1.ibm.com/grid/) includes IBM's grid hardware, operating systems, schedulers, services and client training, and is intended to give businesses a competitive edge by using available resources more efficiently.
- Oracle's grid computing solution (described at www.oracle.com/technology/tech/grid/index.html) lets businesses standardize on modular servers and storage; consolidate servers and storage with Oracle Database (10g) and Real Application Clusters; and automate daily management tasks.
- Sun Microsystems' "Grid Utility Computing" (described at www.sun.com/servers/grid/) is a pay-per-use service that lets users dynamically provision compute power, depending on application requirements. Sun provides access to a standardized grid computing infrastructure that lets you offload your compute-intensive workloads with minimal risk and no capital investment.
- HP is delivering grid-based storage products today that are built according to their "StorageWorks" architecture (described at http://h71028.www7.hp.com/enterprise/cache/125369-0-0-0-121.html). These products either use early versions of smart cell technology or exemplify other design attributes of the architecture, such as single system image management.

The Enterprise Grid Alliance (EGA) is an open, non-profit, vendor-neutral consortium formed to develop enterprise grid solutions and accelerate the deployment of grid computing in enterprises. It consists of more than 30 members, including grid users, vendors and solution providers such as IBM, Oracle, Sun, Intel, HP, DataSynapse, Univa, and Dell. The EGA, as described at www.gridalliance.org, aims to encourage and accelerate movement to an open grid environment through interoperability solutions. It will work on grid computing standards by endorsing and supporting existing specifications, assembling and profiling component specifications, and defining new specifications where needed.

Open Source Software (OSS) is involved in a big way in the development of enterprise grids. Many grid solutions (including IBM's grid service offerings for the enterprise) are currently based on the open source Globus toolkit developed by the Globus Alliance, Argonne National Laboratory, and the University of Chicago and described at www.globus.org. It's a set of software services and libraries for resource monitoring, discovery, and management plus security and file management that facilitate construction of computational grids and grid-based applications, across corporate, institutional and geographic boundaries.

Globus offers grid middleware that mainly runs on the Unix-like platforms. Several open source grid projects have developed user-level middleware that work with the Globus toolkit. One such effort is the Gridbus project at the University of Melbourne, which developed the Grid Service Broker. As described at www.gridbus.org/broker/, the grid service broker supports creation, scheduling, and deployment of computational or data grid applications (including work flows) on enterprise and global grids.

Another open source grid initiative from the Gridbus project is the Alchemi enterprise grid-computing framework that harnesses the power of a network of computers running Windows.



Alchemi: An OS Enterprise Grid Computing Framework

Alchemi, as described at www.alchemi.net, is an open source, .NET-based enterprise grid computing framework developed by researchers at the GRIDS lab, in the Computer Science and Software Engineering Department at the University of Melbourne, Australia. It lets you painlessly aggregate the computing power of networked machines into a virtual supercomputer and develop applications to run on the grid with no additional investment and no discernible impact to users. It's been designed to be easy to use without sacrificing power and flexibility. It supports the Microsoft Windows operating system, which is seen as a key factor in industry adoption of grid computing technology, since more than 90 percent of machines worldwide run variants of Windows.

The main features offered by the Alchemi framework are:

- Virtualization of compute resources across

the LAN/Internet
- Ease of deployment and management
- Object-oriented "grid thread" programming model for grid application development
- File-based "grid job" model for grid-enabling legacy applications
- Web Services interface for interoperability with other grid middleware.

Figure 2 shows the Alchemi architecture, which has three types of components:

- The Manager
- The Executor
- The User Application.

The Manager node is a computer with the Alchemi Manager component installed. Its main function is to service user requests for application distribution. The Manager receives a user request, authenticates it, and distributes the workload across the various Executors that are connected to it. The Executor node is the one that actually performs the computation. Alchemi uses role-based security to authenticate users and authorize execution.

A simple grid is created by installing Executors on each machine that's to be part of the grid and linking them to a central Manager component. The Windows installer setup that comes with the Alchemi distribution and minimal configuration makes it easy to set up a grid.

An Executor can be configured to be dedicated (meaning the Manager initiates application execution directly) or non-dedicated (meaning that the execution is initiated by the Executor). Non-dedicated Executors can work through firewalls and Network Address Translation (NAT) servers since there's only one-way communication between the Executor and Manager. Dedicated Executors are more suited to an intranet environment and non-dedicated Executors are more suited to the Internet environment.

Users can develop, execute and monitor grid applications using the .NET Application Program Interface (API) and tools that are part of the Alchemi Software Developer's Kit (SDK). Alchemi offers a powerful grid thread programming model that makes it easy to develop grid applications and a grid job model for grid-enabling legacy or non-.NET applications.

An optional component is the Cross Platform Manager Web Service that offers interoperability with custom non-.NET grid middleware. Alchemi also comes with a Java API that can be used to develop Java-based clients that need to harness the computing power of an Alchemi grid.

Alchemi is widely used for a variety of applications. It has been used for teaching and setting up test grids and also some serious applications in the commercial world. Some Alchemi-based industrial applications and projects include:

- Large-scale document processing (Tier Technologies, U.S.)
- Natural resource modeling (CSIRO, Australia)
- Asynchronous Excel tasks using Managed XML Linking Language (XLL) (stochastix GmbH, Germany)
- Detection of patterns of transcription factors in mammalian genes (The Friedrich Miescher Institute (FMI) for Biomedical Research, Switzerland)
- Finding the location of a high-frequency radio transmitter using Secure Sockets Layer (SSL) technology (Correlation Systems Ltd., Israel).

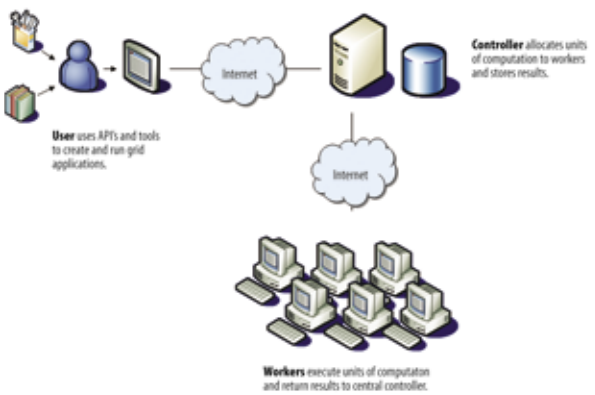


Figure 2: A Simple Alchemi Grid

A Peek Into the Future

The enterprise grid is still in its nascent stages in terms of development and industrywide adoption, but is poised for rapid growth. However, there are issues that preclude the big revolution that the grid promises to bring to IT. Some problems to solve include security, development and wide adoption of standards for representing and executing applications and workflows, resource description, monitoring and management, dynamic service composition and aggregation. There are also issues relating to managing data, intellectual property, developing new software licensing models and their enforcement, representing QoS and formulating and enforcing SLAs that are especially important in a commercial environment. Considerable research is under way in these areas and the standards are constantly evolving. Finally, before grid computing becomes ubiquitous, a sustainable business model has to be developed so all parties obtain value from adopting grid technologies.

It's widely believed that the grid of the future will be based on SOA and software and hardware will be available as a utility with demand and supply regulated by the concept of an economic market—just like it works for any other utility such as electricity, telephone, and water. ●

Rajkumar Buyya, Ph.D., is director of the Grid Computing and Distributed Systems (GRIDS) Laboratory within the Department of Computer Science and Software Engineering at the University of Melbourne, Australia. He has pioneered the economic paradigm for service-oriented grid computing and demonstrated its utility through his contribution to conceptualization, design and development of cluster and grid technologies, such as Gridbus, that powers many emerging eScience and eBusiness applications.
e-Mail: raj@cs.mu.OZ.AU

Krishna Nadiminti is a research programmer for the GRIDS laboratory, Department of Computer Science and Software Engineering, the University of Melbourne, Australia.
e-Mail: kna@cs.mu.OZ.AU

BY DAVID E. GOULD

GPL Prediction Latitude

Due dates are tricky things. This column was due right before the release of the first draft of the GPL3 GNU Public License Version 3 (GPL3). However, given the importance of the GPL to the open source community, and the sheer number of discussions that will center around the GPL3 in 2006, I couldn't pass up the opportunity to talk about it, even though by the time you read this, you will have the advantage of having seen the actual first draft of the GPL3. I will not have seen it, and as a transactional lawyer, I'm not usually given to prognostication (I leave that to the litigators), so I ask you to keep that in mind as you judge the following predictions.

From my experience, the first draft of an agreement often addresses the large issues, but it's normal to see a myriad of smaller ones not addressed, and in some drafts, to further find missing definitions or concepts, placeholders, and even internal contradictions. Even though this first draft of the GPL3 was in the works for more than a decade, it wouldn't be fair to the Free Software Foundation (FSF) to assume their first draft would be entirely free of all of those characteristics.

However, the GPL3 Process Definition (released Dec. 1, 2005) did show that, in addition to spending time on the drafting, the FSF had also given considerable thought to the process by which the GPL3 would be extensively commented upon, and therefore, I do anticipate that the first draft of the GPL3 will have addressed quite a large number of the "smaller" issues from the GPL2. Again, the FSF seemed to strongly imply this in the GPL3 Process Definition, and I also believe the FSF did seek to use the lengthy drafting time to clean up as many of these issues as possible.

For example, I predict that GPL3 will have to deal with the "enterprise" concept in a much clearer manner than the Annotations of GPL Version 2 did, trading the use of the ambiguous "organizations (including companies)" for a much clearer definition (these definitions will appear in the main text of the GPL3). My thoughts on that are that the enterprise would mean the GPL3 licensee and its affiliates (further defined as all entities controlling, controlled by or under common control with), and the usage rights flow not just to the entities, but to their respective employees, independent contractors, agents, and other third parties utilizing the GPL3 licensed program on behalf of the licensee or its affiliates.

Hidden in the enterprise comment of mine is yet another prediction: The GPL3 will make much clearer the fundamental concept of free usage of the GPL3 licensed software. I realize the GPL3 Process Definition stated, "It goes without saying that people have the freedom to run a program under the GPL," but I continue to believe it will need to be said (and in a much clearer fashion than was the case in GPL2), so I make this a longer term 2006 prediction: The right to use will be explicit in GPL3 (if not in the initial draft, then as a result of the comment process).

One more related prediction: The concept of "distribution" will be much better defined in the GPL3. For example, the confusion perpetrated by the Frequently Asked Questions of the GPL2 (in which intra-"organization" duplication wasn't distribution, but transfers to "other organizations or individuals" and even "contractors for use off-site" was distribution). While there has been some argument from anti-OSS sources that some of these concepts were intentionally ill-defined in the GPL2, I continue to read the Process Definition as a serious commitment by the FSF to reduce and not retain the level of ambiguity in GPL3.

From my experience, the first draft of an agreement often addresses the large issues, but it's normal to see a myriad of smaller ones not addressed …

My final prediction: A future column will start with a discussion of why these predictions didn't come to pass, but quickly move on to discuss several of the entirely new issues in GPL3, and what to do about those. ●

David E. Gould is an associate in Buchanan Ingersoll's Technology Transactions Group. His practice focuses on the selection, negotiation, acquisition, and disposition of technology systems. He also writes a Web column on associate life issues.
e-Mail: gouldde@bipc.com

Enterprise Readiness

Dominion Insurance Services Inc., Alpine, UT, had been using a proprietary Database Management System (DBMS) for years. Then Dominion started to grow. "The database vendor was changing the costs on us by processor and by users. It got to the point where we wanted to use the database over the Web and they were going to hit us with a minimum six-figure fee," says Larry Hilton, president.

The year was 1998 and open source database technology hadn't yet been widely recognized as a serious enterprise option. Still, the company, aided by consultants, switched to PostgreSQL.

"PostgreSQL has been very stable and we like being able to tailor it to our exact needs," says Hilton. In addition, you can't beat the cost.

Now the enterprise open source database market is heating up. It has experienced a surge of activity just in time to meet sudden demand.

"Organizations will double the number of databases they run in the next five to seven years," says Noel Yuhanna, senior analyst, Forrester Research, Cambridge, MA. That means managers everywhere will be revisiting the DBMS question. And this time they will find several viable options.

Although few organizations are likely to rip and replace their existing proprietary databases simply because a lower cost open source alternative is available, they may turn to an open source DBMS as they add new databases. Not every application needs a full-blown proprietary enterprise DBMS with all the expense and effort it entails. For enterprise IT managers, open source databases have become a viable option. The selection options in the open source enterprise database segment are varied; the vendors and open source communities are adding increasingly sophisticated capabilities that rival the big proprietary databases. Open source databases give managers alternatives that will do almost any database job with reduced costs.

Surge of DBMS Activity

"There are more than a dozen open source databases today, and that number is increasing," reports Yuhanna. "Five key products dominate the market: Berkeley DB, Cloudscape/Derby, Ingres, MySQL, and PostgreSQL."

However, the latest activity promises to shake up what has been a pretty static market.

Venture capitalists, for example, backed Enterprise DB, an open source database specifically intended to attract customers from proprietary database heavyweight Oracle. Just in case the fledgling Enterprise DB actually gains traction, Oracle responded by acquiring an open source database of its own, Innobase, from a Finnish company.

About the same time, CA decided to divest itself of Ingres. CA had previously turned Ingres, which had been one of the early proprietary enterprise databases, into open source. Venture buyout specialists are keeping Ingres as an open source database and plan to offer services on top of the software, according to the Ingres announcement. In its proprietary heyday, Ingres competed directly against Oracle, Sybase's SQL Server, and IBM's DB2, today's leading proprietary enterprise databases.

IBM offers its own open source database, Cloudscape, although it's definitely not designed for enterprise computing. It gave Cloudscape to the Apache community as Derby.

"Cloudscape could support 20 to 30 concurrent sessions, but everything would have to fit on one disk drive, and it isn't multithreaded," says Kevin Foster, IBM manager for Cloudscape and DB2.

While it's not enterprise-class, IBM recommends that independent software vendors use Cloudscape in their products to start and move up to DB2 when their customers need to scale.

Open Source Database Market

All industry analysts and observers agree on one thing: The open source database market is growing. Forrester estimates the current open source database market, which consists of new licenses, support and services, will exceed $1 billion by 2008.

Open source database deployments were up 20 percent in 2005, according to Evans Data. By contrast, Gartner, in published reports, projects the worldwide DBMS software market, including proprietary and open source products, to grow at a Compound Annual Growth Rate (CAGR) of 6.6 percent, which would take the market to $13.2 billion in new license revenue by 2009. By then, open source databases, which are growing at a faster CAGR than the market as a whole, will likely account for more than 10 percent of the market in revenue terms, which isn't a trivial achievement.

In his report titled "Open Source Databases Come of Age," Yuhanna identifies six factors that are driving enterprises to open source databases:

- Low acquisition cost
- Strong support from the open source community
- Reduced maintenance costs
- More hardware and software options
- Access to the source code
- Avoidance of vendor dependence.

# of Open Source Databases

By Alan Radding

Open source databases aren't without drawbacks. For example, several popular packaged enterprise applications are intended to run with the leading proprietary DBMS only, not open source DBMS. Additionally, open source technical support may not be as extensive or global as that provided by the large proprietary vendors. Finally, certain costs, such as administrative overhead, remain the same whether the DBMS is proprietary or open source.

Open Source Database Landscape

Although the five products noted previously top the open source DBMS charts, the landscape is shifting as new players, such as Enterprise DB, arrive and existing products are enhanced.

MySQL appears securely entrenched as the market leader. The database boasts of major enterprise customers such as Sabre, the airline reservation giant, and The Weather Channel. The cost for MySQL Network, the version intended for enterprises, ranges from $500 to $5,000—still far below the lowest costs for even Microsoft's SQL Server.

In fall 2005, MySQL introduced the latest version with a host of enhancements, including read-only and updatable views, stored procedures, row-level triggers, server-side cursors, and a data dictionary (metadata repository). It also provides a federated DBMS engine, which allows for the creation of one logical database from tables residing in multiple, remote databases. MySQL 5.0 runs on Linux, Windows, Solaris, Mac OS X, FreeBSD, HP-UX, IBM AIX, and other operating systems and is available under a dual licensing model, either open source GPL or a commercial license.

Diabetech LP, Dallas, a company that helps people manage their diabetes through information collection and management, has been relying on MySQL for four years. After evaluating PostgreSQL, the company opted for MySQL.

"MySQL's engine was most attractive," says Eric Link, Diabetech CTO. "It offered a lot of flexibility." The company makes extensive use of JasperReports, an open source business intelligence product, in conjunction with MySQL and JBoss to deliver its core information management and reporting capabilities.

PostgreSQL has matched and beaten MySQL feature for feature for years. It has included triggers and stored procedures for years and supports high-availability clustering, reports Josh Berkus, a core team member of the PostgreSQL community. It has focused on Online Transaction Processing (OLTP) more than MySQL and has had success targeting Oracle users for both transaction processing and data warehousing.

"Both MySQL and Postgre are a key part of the LAMP (Linux, Apache, MySQL/Postgre, PHP) stack," says a senior database engineer now working at IBM. "MySQL has gotten all the buzz, but Postgre has better performance and fewer gotchas. I've used it for both OLTP and decision support. In my experience, Postgre scales bigger. Postgre is architected to be big."

The newest guy on the block is Enterprise DB. Based on PostgreSQL, Enterprise DB is aiming squarely at the Oracle market.

"What we've added is Oracle compatibility," says CEO Andy Astor.

The company has experienced tens of thousands of downloads since its August 2005 launch and expects it to be used for enterprise OLTP applications.

The company allows a free download of the source code and charges for support, with subscription costs running $1,000 to $5,000 per year per CPU—not exactly cheap but considerably less than a proprietary DBMS.

Enterprise DB promises a complete enterprise-class DBMS with stored procedures, locking, and concurrency control, unimpeachable security and reliability, compatibility with open standards, and scalability.

"Either you get 100 percent reliability or it gives you an error message," says Astor, "so you know if the transaction went through."

The other players in the open source database arena are niche players such as Cloudscape or low-profile products such as Ingres. But as interest in open source enterprise databases grows, expect to see more action even among these players.

Today, you can get an open source database that can nearly match the big proprietary databases feature for feature. They're also gaining the scalability and reliability that have been hallmarks of the proprietary DBMS.

Says the database engineer, "You might not use it for an airline's OLTP reservation system or a large brokerage firm that handles thousands of transactions a minute, but for anything else, the open source databases are pretty darn good."

Considering you can get "pretty darned good" for thousands of dollars less per server per year than a proprietary DBMS, most businesses would consider open source databases nothing short of outstanding. ●

Alan Radding is a freelance writer based in Newton, MA. He specializes in business and technology.
Voice: 617-332-4369
e-Mail: alan@radding.net
Website: www.technologywriter.com

BY ROBERT LEFKOWITZ

What's Your Recommended Daily Allowance of Open Source?

The Food and Drug Administration (FDA) has just updated its dietary guidelines to increase the number of daily servings of fruits and vegetables to five to 13 servings a day (the exact amount varies based on activity level, age, and other factors). Unfortunately, studies by the U.S. Department of Health and Human Services (DoHHS) show that 42 percent of Americans eat less than two servings a day—and the average is between three and four servings.

I was recently reminded of this survey when reading the results of a survey conducted by Optaros (an enterprise open source consultancy) on the use of Open Source Software (OSS) in U.S. organizations. The survey reported that 87 percent of organizations were using OSS, and that 42 percent were using open source content management systems. As the 42 percent figure appeared in both surveys, the coincidence drew my attention.

The interpretation of the DoHHS survey is that the 42 percent figure is appalling—it's a call to action to galvanize Americans to triple their intake of fruits and vegetables. Out of concern for our collective health, the National Cancer Institute has been waging a decades-long campaign called the 5-A-Day challenge to drive toward that minimum healthy level. By comparison, what's lacking in the open source survey is a value judgment about whether those 42 percent of companies are using too much, too little, or just the right amount of OSS. Granted, different companies have different needs (as different lifestyles have different dietary requirements). It should certainly be possible to establish a range for "minimum and optimal requirements" for open source usage that would cover most American organizations—something like "five to 13" servings daily.

If open source is a good idea, how much open source do you need to achieve those health benefits? Here's another way to interpret the Optaros survey: If 87 percent of American organizations are using OSS, then the untapped market is only the 13 percent of remaining companies. Another 15 percent market growth in open source enterprise software and the saturated market will tail off—leaving many disappointed investors. Perhaps the lesson of this survey is that those 87 percent use open source as (just guessing) 1 percent of their technology mix, and a healthy diet would require (just making something up) 33 percent of their technology mix. That would leave plenty of room for growth.

The Free Software movement isn't reluctant to propose a target—100 percent of software should be open source. To continue stretching my analogy, they're "vegetarians." The open source community is more omnivorous—believing a balanced diet is more appropriate. I have yet to see, however, a proposal for a Recommended Daily Allowance. If we're going to make rational decisions, we're going to need to wrestle with quantification. There are at least three questions confronting us:

1. Which things are or aren't fruits and vegetables? With all the hybrid business models around hybrid open source, which projects or products have the nutritious elements that would qualify them as healthy open source, and which things are Gummy Bears—fruit flavored, but aren't actually fruit?
2. How much is a "serving"? How does one measure the "amount" of OSS being used?
3. How many servings are healthy?

Working backward on that last question, let's estimate the appropriate balance between OSS and proprietary software. Assume the answer is on the order of 10 percent OSS. Then it would be fair to say that OSS should rarely command the attention of the CIO or CTO. Another way to express that thought is that 90 percent of software should be proprietary. To get a few minutes on the executive agenda, the appropriate mix would need to be at least 25 percent or maybe even closer to 40 percent OSS. That still means the majority of software should be proprietary, but a noticeable amount should be open source. Given the way IT budgets are distributed, to achieve that level of impact, we would need to find relationships between open source, development and maintenance, infrastructure and applications, services and outsourcing, salaries, benefits, and—also—software licensing. I plan to look for and explore those relationships.

For a healthy IT organization, the appropriate level of open source usage should be 33 percent, which is to say, you should be using twice as much proprietary software as OSS. In what percentage of U.S.-based companies does open source account for about 33 percent of software? How many use more? How many use less? That's the survey I'd like to see. ●

Robert Lefkowitz has spent more than 20 years being a contrarian inside large IT organizations. His first open source job dates back to the late '70s as the public software librarian for a timesharing company.
e-Mail: r0ml@mac.com

BY DENNY YOST

Training Funding Partners Achieves Improved Scalability and Nationwide Expansion Using an Open Source ERP Solution

Companies of all sizes are learning about the benefits of using open source solutions to run their businesses. One such company is Training Funding Partners (TFP), a California-based company helping *Fortune* 100 to *Fortune* 2000 companies locate, secure, and manage millions of dollars in workforce training grants awarded by states for companies to train their employees. To qualify for and receive training reimbursement funds, these companies are required to complete detailed applications with little tolerance for errors. Once a company is awarded a reimbursement contract, strict training tracking procedures must be followed and reported to each appropriate state. This can easily make managing training reimbursement funds an overwhelming job.

TFP had successfully supported its CA-based clients with its original, highly manual workflow process, but the company wanted to cost-effectively scale the business beyond California. This would enable them to deliver more services to current clients and reach a broader range of potential clients nationwide. To do this, TFP needed a way to correctly process each state's complex program rules inherent in the company's business services, automate the application of these rules, ease the management of secured reimbursement funding contracts, and free staff to spend their time in areas that require their knowledge and expertise. The solution was to find an Enterprise Resource Planning (ERP) system that would meet their needs.

To ensure a successful outcome for such a mission-critical project, TFP first aligned itself with Idalica, a knowledgeable consultative software services company, to help identify and implement a new solution. "We knew what we needed for an outcome, we knew it wouldn't be an off-the-shelf solution, and we knew from previous experience that Don Lagwig could help us," says Mark Coleman, TFP's CEO.

The search for an ERP solution began by reviewing the offerings of well-known proprietary systems. "We reviewed the ERP offerings from Microsoft, SAP and Oracle," said Don Ladwig, CEO at Idalica. "While these are all good systems, TFP needed a solution that could easily be customized to meet the needs of the company and be fully integrated into TFP's business processes. Proprietary ERP vendors have service teams that perform this level of customization; however, for most companies, the cost of having the vendor provide customization is equal to or greater than the cost of the product license. Then, there's the cost of ongoing maintenance. These costs can easily extend the ROI for such a system to five years or more. TFP needed a two-year ROI. This is what drove us to consider the open source ERP system from Compiere."

The Compiere ERP & CRM solution offered TFP three key benefits: source code access for easy customization, a comprehensive solution, and affordable pricing. "Compiere had everything TFP needed in an ERP system," Ladwig continues. "It was created to operate as a mission-critical system, meaning actions are built into the product to automatically perform certain safeguards. The Compiere source code is also available to anyone, the product license is free, and ongoing maintenance is reasonably priced. After doing some initial tests and reviewing TFP's needs against the Compiere product, we estimated TFP would have a two-year ROI. We downloaded the product and started working."

Today, the new Compiere system is delivering big benefits to TFP. "So far we've been able to accomplish the goals we wanted to achieve by replacing our old system," comments Coleman. "The customized Compiere system also gives TFP a huge competitive advantage in the market, which we didn't set as a primary goal. We've successfully scaled our business, enabling us to expand nationally without making significant staff increases, efficiencies throughout the company are becoming more visible every day, and we can provide our clients with more services than ever before. It's especially easy to see how small to medium-size companies can gain significant benefits from an open source ERP system like Compiere."

For more information, please contact Compiere at www.compiere.org/contact.html or Idalica at dladwig@idalica.com.

how to evaluate

Community Support for an Open Source Content Management System:

PART II

Evaluating the Life of the Community

by boris kraft

In Part I, we saw that the community is essential to the success of any Open Source Software (OSS) project—and to the successful deployment of the software within your enterprise. We identified the major groups that compose a community—users, contributors and committers—and described the role they play in the community. We discussed how you should take a good look at the community of a project instead of deciding on technical issues alone. That's because the community provides free support, examples, secures a project's future, and ensures high software maturity. Community is orthogonal to all the other purchasing issues—except the legal ones—which is why we look at evaluating the community that supports an OSS system.

Ways to Evaluate Community: Is it Quantity or Quality You Are Looking For?

On Nov. 16, 1532, Francisco Pizarro lead a group of 168 Spanish solders to attack Atahuallpa, the ruler of the largest, most advanced state in the New World. Situated in the Peruvian highland town of Cajamarca, Atahuallpa was protected by 80,000 of his own soldiers and ruled over an empire of millions of subjects. Pizzaro was unfamiliar with the terrain, far away from home, with no outlook of timely reinforcements. Would he and his community of 168 soldiers have any chance to see the dawn of the next morning with their heads still on their shoulders? Put in other words—what is the relevance of quality vs. quantity?

The community is made up of people; the naïve way to judge its strength is to count its size. Straightforward as it sounds, hardly any project simply publishes community

statistics on its home page. Even if they would, we know writing killer software isn't a numbers game—it requires vision, skill, determination, and experience at the very least. These are issues of quality, not quantity.

Thus, the aspiring community evaluator faces two problems: a) it's hard to count the people, and b) it's of limited use. Nevertheless, we will show you how to get at the numbers and learn a lot about the quality of the project (and the people involved) while doing so.

As there is hardly a project that openly states its community's size, we will take a detour and look at the artifacts they produce: e-mail traffic, issues reported, and code written.

We will start by looking at the most prominent artifacts the user group produces: support questions and answers. Then we will gauge the group of contributors by checking available bug reports, documentation, and localization. Finally, we will examine the source code to learn about the quantity and quality of the project's committers.

Mail and Forum Support

Commonly, OSS projects have at least one mailing list for the user community and another for the developer community. Often, there's also a list that tracks bug reports (issues), a list that tracks the source code changes, and an announcement or news list. And, instead of or in addition to lists, forum software lets users post questions anonymously.

For an impression of what's happening on a specific list, browse its archives or try to get some statistics about the activity on the list. That's not always straightforward, as each project has its own structure, different software for lists or forums, and different statistics.

Before you run out and buy a stack of books on the subject, here's a life-saver: Several services on the Web provide gateways to project mailing lists, usually in the form of a Web interface. One such example is gmane.org. Gmane provides many advanced features, the most valuable being that it:

- Lets you find the lists that exist for a project (if it monitors them)
- Gives detailed statistics about the traffic on the list
- Lets you browse and search the list.

For instance, there are three monitored mailing lists for the Magnolia CMS on Gmane—an announcement list, user list and developer list, together with the total of messages that Gmane has tracked for the respective lists (see Figure 1).

Figure 1: Three Monitored Mailing Lists for the Magnolia CMS on Gmane

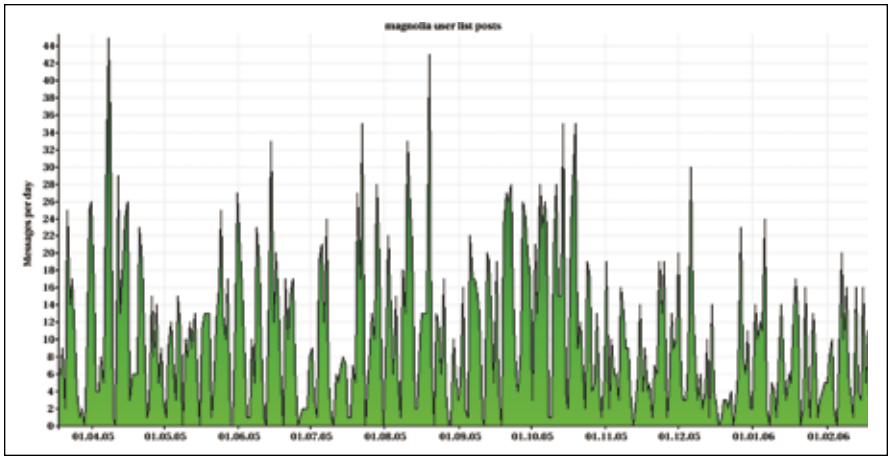


Figure 2: Posting Rate on gmane.com for the Magnolia User List

At this level, we don't know when Gmane started tracking a list and can't really say much about the project except that it exists and at some time generated mail traffic.

If you click through to the list overview, you'll get an overview of the traffic on the list and immediately see when tracking started and how many messages are posted daily. Again using the Magnolia Content Management System (CMS) user list as an example, we see that 10 to 20 messages per day are sent to this list on average, peaking out at 60 messages on rare and very busy days.

Gmane will also show statistics of how many people are posting to the list daily (unfortunately, this feature seems broken), how many subjects are being discussed daily, and how many messages are reportedly spam (see Figure 2). The number of unique subjects per day gives a good indication of quality. If the ratio of posted to unique messages is high, it's more likely that topics are discussed in-depth, usually signaling interesting subjects. Conversely, if the number of unique subjects matches the number of messages (the ratio would be one), apparently no one is answering the questions posted—even if hundreds of messages are posted each day, this would be useless for you.

Finally, the mail archives on Gmane can offer an impression of what types of questions are commonly or currently being asked. Spending about an hour with the archives of a single project should give you an impression of the quality of the community support and the product. You will get a feel for the number of people involved in the discussions, find out the key people, and identify the most common problems.

Mailing lists and forums only tell a small part of the story. If a CMS is hard to use or install, users might join the list only to get their immediate problem solved and then vanish. Thus, no real community is established. Think of a town where everybody moves on after a year or two. In such a place, no sense of "us" is established, no common values are developed, and no traditions are followed. Similarly, if a project consists of only one or two people who answer all the questions on a mailing list, then there's no existing community, no matter how much traffic the list carries. More difficult to spot is the case where others do answer, but only for a short time. This typically happens

when potential contributors work with a product only for the duration of a single project. When their professional interest moves on, so do they.

A good, lively user community is the foundation for the success of an OSS project, since enthusiastic users will contribute something back to the community. They may, for example, document how they've solved a specific challenge. In doing so, they become contributors.

Bugs, Documentation, and Localization

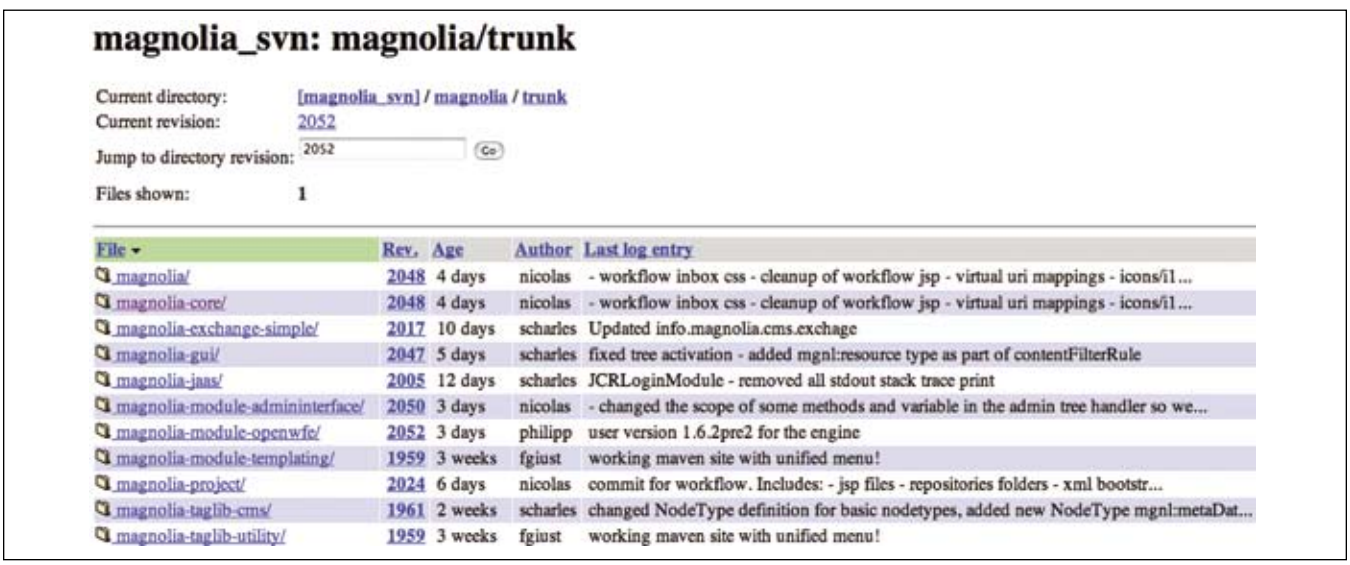
Some of the most important artifacts contributors produce are bug reports (and fixes), documentation, and localization. Looking at these provides a clear measure of the activity and size of this group.

OSS projects will most likely not spend their financial resources to get professional translations, since this is a part that project contributors can easily provide. So, the easiest, quickest way to gauge the contributors is to check the number of translations that exist for the CMS of your choice. The more languages a CMS has been translated to, the bigger its group of contributors. SourceForge lists the available languages of each project in the project's summary page; so does freshmeat, a directory of software. Some CMSes won't be listed there and you have to look for available translations on their home page, but generally, that should be straightforward.

If you wish to have better insight into contributors, evaluate the data a project's issue tracker provides. Besides the obvious—documenting open and fixed issues—the issue tracker records a lot of meta information that gives a pretty clear picture of contributor activity. An issue tracker tells the number of people using the system, number of issues posted within a certain time, the duration until issues get fixed, and the number of people providing patches.

T	Key	Summary	Affects Version/s	Fix Version/s	Assignee	Pr	Status	Res	Created	Updated	Due	Reporter
	MAGNOLIA-616	src/main/info/magnolia/cms/taglibs/SetNode.java Does not implement a proper map for JSP page	2.1.3		Boris Kraft		Open	UNRESOLVED	11/Jan/06	11/Jan/06		Ramon Buckland
	MAGNOLIA-615	content items fail to achieve persistent state, concurrent access 5+ users	2.1.3		Philipp Bracher		Open	UNRESOLVED	20/Dec/05	23/Jan/06		Jochen Fliedner
	MAGNOLIA-614	going from edit page to admincentral results in inability to activate content	2.1.1		Philipp Bracher		Open	UNRESOLVED	20/Dec/05	31/Dec/05		Boris Kraft
	MAGNOLIA-613	Import of XML Data makes problems	2.1.3		Sameer Charles		Open	UNRESOLVED	19/Dec/05	19/Dec/05		Gadget
	MAGNOLIA-612	i18n in Dialogs	3.0 RC1	3.0 RC1	Philipp Bracher		Open	UNRESOLVED	09/Dec/05	09/Dec/05		Stefan Grünig
	MAGNOLIA-611	server/i18n Node	3.0 RC1	3.0 RC1	Philipp Bracher		Open	UNRESOLVED	09/Dec/05	09/Dec/05		Stefan Grünig
	MAGNOLIA-610	magnolia.war	3.0 RC1	3.0 RC1	Philipp Bracher		Open	UNRESOLVED	09/Dec/05	09/Dec/05		Stefan Grünig
	MAGNOLIA-609	javascript-encryption of mailto-links	2.0 Final	3.0 RC1	Philipp Bracher		Open	UNRESOLVED	09/Dec/05	09/Dec/05		Caecilia Schindler
	MAGNOLIA-608	MAGNOLIA-587 ↳ Spring integration?			Boris Kraft		Open	UNRESOLVED	09/Dec/05	16/Jan/06		Philipp Bracher
	MAGNOLIA-607	When you click on the template selection for an already existing page is automatically changes the template to the first one in the list.			Philipp Bracher		Open	UNRESOLVED	01/Dec/05	01/Dec/05		Philipp Bracher
	MAGNOLIA-605	Create a help property in the dialogs	3.0 RC1 2.1.4		Philipp Bracher		Open	UNRESOLVED	01/Dec/05	03/Dec/05		Tom Wespi
	MAGNOLIA-604	Build documentation refers to svn trunk instead of magnolia2.1 branch	2.1.4		Philipp Bracher		Open	UNRESOLVED	30/Nov/05	01/Dec/05		Markus Strickler
	MAGNOLIA-603	Export fails due to "invalid XML character" found in content	2.1.3		Sameer Charles		Open	UNRESOLVED	29/Nov/05	29/Nov/05		Thomas Duffey
	MAGNOLIA-598	Option to set lifetime on allowed cache mappings			Sameer Charles		Open	UNRESOLVED	14/Nov/05	30/Nov/05		Thomas Duffey
	MAGNOLIA-595	Search Indexing is failing with a null pointer exception	2.1.4 2.1.3		Philipp Bracher		Open	UNRESOLVED	10/Nov/05	30/Nov/05		Craig Boxall
	MAGNOLIA-593	MAGNOLIA-587 ↳ Command architecture	3.1	3.1	Philipp Bracher		Open	UNRESOLVED	02/Nov/05	09/Dec/05		Philipp Bracher
	MAGNOLIA-591	Bad characters in dialogs.	2.1 Final		Philipp Bracher		Open	UNRESOLVED	01/Nov/05	30/Nov/05		Miklos Tisza
	MAGNOLIA-590	Cross Site Scripting Vulnerability (XSS) in Search template			Philipp Bracher		Open	UNRESOLVED	01/Nov/05	18/Dec/05		Oliver Lietz
	MAGNOLIA-589	MAGNOLIA-587 ↳ Admin interface: XML-RPC vs SOAP	3.1	3.1	Boris Kraft		Open	UNRESOLVED	31/Oct/05	09/Dec/05		Philipp Bracher

Figure 3: Issue Tracking in Open Source



magnolia_svn: magnolia/trunk

Current directory: [magnolia_svn] / magnolia / trunk
Current revision: 2052
Jump to directory revision: 2052 Go
Files shown: 1

File	Rev.	Age	Author	Last log entry
magnolia/	2048	4 days	nicolas	- workflow inbox css - cleanup of workflow jsp - virtual uri mappings - icons/il...
magnolia-core/	2048	4 days	nicolas	- workflow inbox css - cleanup of workflow jsp - virtual uri mappings - icons/il...
magnolia-exchange-simple/	2017	10 days	scharles	Updated info.magnolia.cms.exchage
magnolia-gui/	2047	5 days	scharles	fixed tree activation - added mgnl:resource type as part of contentFilterRule
magnolia-jaas/	2005	12 days	scharles	JCRLoginModule - removed all stdout stack trace print
magnolia-module-admininterface/	2050	3 days	nicolas	- changed the scope of some methods and variable in the admin tree handler so we...
magnolia-module-openwfe/	2052	3 days	philipp	user version 1.6.2pre2 for the engine
magnolia-module-templating/	1959	3 weeks	fgiust	working maven site with unified menu!
magnolia-project/	2024	6 days	nicolas	commit for workflow. Includes: - jsp files - repositories folders - xml bootstr...
magnolia-taglib-cms/	1961	2 weeks	scharles	changed NodeType definition for basic nodetypes, added new NodeType mgnl:metaDat...
magnolia-taglib-utility/	1959	3 weeks	fgiust	working maven site with unified menu!

Figure 4: ViewCVS Showing the Modular Structure of the Magnolia Project

There are three commonly used issue trackers in the OSS community—the one provided by SourceForge, open source Bugzilla, and proprietary JIRA (which provides a free license to OSS projects). Both Bugzilla and JIRA let you export a list of bugs and make it trivial to find out how many people have reported issues and who these people are. SourceForge makes it a bit harder; you can't get a listing of, say, all open bugs, but instead must page through them 25 issues at a time. But even SourceForge lists the reporter (i.e., the person who reported the issue), which is all we want to know for a start (see Figure 3).

While the number of localizations and a look at the issue tracker will provide significant insight into the strength of the group of contributors, examining the available documentation—while hard to measure—will tell you how mature a project is.

Documentation is an essential part of any software. Documentation between commercial products and open source products has significant differences. While proprietary software is accompanied by thick manuals, it often fails to provide insight into the development process, Application Program Interfaces (APIs), open or solved issues, and examples from other users—issues where OSS has its strong points. OSS generally also implies open communication; you have access to issues and how they've been solved. You can see what's happening in real-time (source code repository access) and the development of the project is freely discussed on the mailing lists. All this is an important part of the documentation of a product. Next, the community often participates on a wiki—a Website that anyone can edit. This is often a good place to find specific solutions, code examples, or how-to's.

Source code is, in some sense, the ultimate documentation of a product. It might not be easily accessible for an end user, but if documentation and behavior of a product differ, the code is always right. In terms of enterprise CMS needs, access to the code and its API documentation are extremely valuable once you start to integrate your company's information assets into your intranet CMS, or wish to write custom modules that provide specific functionality otherwise unavailable. In these cases, good API documentation is essential, and you should examine it to see if it provides the level of information you need. The API is often available for download or even online (check the developer section of the project's home page).

Documentation needs will differ, depending on what you wish to achieve with the implementation of a CMS in your enterprise. Sometimes a user manual is best written by your company's staff after you've customized the system. That said, good documentation in the classical, proprietary sense is something you definitely will want to have an eye on when evaluating your next CMS.

Code

Source code is often the first thing that's released by an OSS project—it's an important artifact you can use to judge the quality and liveliness of the community.

While generally no direct data is available about the size or demographics of the user group and the contributor group, OSS projects always list their core members. For instance, many projects are hosted on SourceForge, where the project summary displays the number of developers, and detailed views show you who they are and on what other projects they're working. Check the source code and code repository for insight about the number of committers—and with some luck, even tools to measure quality are provided.

Source code is usually available in a versioned source code repository—common is either Concurrent Versioning System (CVS) or Subversion (SVN), both OSS implementations themselves. The repositories let you check how many people work on the code, who has commit rights, and when the last changes were made. If you're lucky, the project you evaluate has ViewCVS installed. ViewCVS provides Web-based access to source code repositories and lets you see, at a glance, when the last change was made and by whom.

Modern software management tools such as Maven (OS) generate complete documentation that includes all sorts of metrics. It's possible to see cyclic dependencies or code that has changed numerous times. Thus, it's easy to find out the maturity and quality of the code.

You might want to review how modular the code is. Maven 2 provides the new functionality to modularize the source code and build each module separately. A modular source code allows for far more community participation, since a possible contributor doesn't need to understand the complete source code—it's enough to have well-defined APIs and work on a single module (see Figure 4).

Conclusion

We have examined several artifacts a community produces and that can help you gauge the liveliness of a project's community. Available translations, mail and forum traffic, documentation and source code are important output that provides deep insight into the healthiness of a community.

This series showed us that the community is composed of people who share a common interest. Once you start using an OSS product, you'll share some of that interest, and should think about taking a more active role in the community. Giving is better than taking, and investing in an OSS project will provide your enterprise with significant benefits because you can:

- Directly influence the direction of the project
- Save significant costs compared to proprietary, in-house development
- Ensure the project's survival and increase its viability
- Gain visibility and become a much more attractive employer for some of the most talented programmers in the world.

If you consider how hard it is to hire a good programmer, and how much harder it is to keep him, you will quickly realize the benefits of the last point alone outweigh the cost of active participation.

As we've seen, both quantity and quality have to be considered when you need to judge the community of an open source CMS. With that in mind, how do you think Francisco Pizzaro and his 168 men fared on Nov. 16, 1532? Not only did he survive the day, he and his 168 men captured Atahuallpa within hours, extorted from him the largest ransom recorded in history, and won a decisive victory.

Consider this a vivid illustration of quality outweighing quantity—in this case, by a factor of 500. ●

Boris Kraft is a consultant and software developer. He is the strategic leader and community manager of Magnolia, an open source enterprise content suite, originally developed by his company, Obinary. He has been writing software for 25 years and lives in Switzerland with his wife and two kids.
e-Mail: boris.kraft@obinary.com
Website: www.obinary.com

BY JIM ZEMLIN

The Latest Standards Battlefield

By now most of you have heard of the State of Massachusetts decision to publicly endorse the Open Document Format (ODF) for its public records. It's been a hot topic in IT media, especially those concerned with open source and open standards, and has even become an issue (albeit a small one) in the upcoming governor's election. You also may be aware that I closely watch the intersection of open source and open standards and think this combination is vitally important for the protection of end users' computing assets.

In September 2005, the State of Massachusetts published its endorsement of ODF and its rejection of Microsoft's proprietary XML format used in its Office applications. It wasn't a rash decision; it was a product of more than two years of investigation, including extensive discussions with Microsoft itself. The State certainly didn't preclude Microsoft from competing for business; it merely said it wanted an open standard to be supported by all future technology purchases.

So what was the State of Massachusetts trying to achieve with this mandate? Let's go to their official policy statement:

- "Effective and efficient government service delivery requires system integration and data sharing.
- Technology investments must be made based on total cost of ownership and best value to the Commonwealth. Component-based software development based on open standards allows for a more cost-effective, build once, use many time's approach.
- Open systems and specifications are often less costly to acquire, develop, and maintain and don't result in vendor lock-in."

In standards-adoption, government agencies are early adopters. They have a fiduciary responsibility to the citizens who support them, and standards have overwhelmingly proven successful at reducing the cost of procuring and maintaining technology. Interoperability and access to information is also vitally important to public agencies. Standards such as ODF (or the Linux Standard Base [LSB], for that matter) ensure access to information now and in the future, something that proprietary formats, despite their ubiquity today, simply can't guarantee.

What exactly do we mean by open standards? Basically, an open definition requires unfettered access to the formation and implementation of that standard. This means the standard is developed in a public forum by a wide variety of individuals and stakeholders. The standard must be publicly available for implementation *without restriction*. This doesn't mean all implementations of ODF (or other open standards) are open source; in fact, there are many proprietary implementations of ODF today.

Let's look at the competing standard to ODF, Microsoft's XML format. During Microsoft's vitriolic fight against the decision to support ODF, the company submitted its document standard to ECMA, a standards organization based in Europe, and pledged to go further than it ever has to make it "open." While all the details of ECMA's definition of the Microsoft standard remain to be seen, a few things have already come to light. It was certainly not created in a public forum with a variety of inputs; in fact, reports state it will be impossible for anyone to improve the standard (an important point). While some in the open source community have applauded the move to ECMA, there have been conflicting reports that Microsoft, as it has in the past, will impose Draconian intellectual licensing restrictions on anyone making use of its standard. Perhaps the pressure from Massachusetts and its resulting press coverage will change this. If so, it will be a victory for open standards and, hopefully, a sign of things to come from Microsoft and other vendors.

One of Microsoft's main criticisms of ODF is that it may not have a guaranteed future; if key vendors such as Sun stop supporting it, the State will have to re-save and re-format its documents yet again. While this is a fair criticism, the proponents of open standards (and open source) say you're far more protected by a broad coalition of vendors and communities united around a truly open standard than by a single company, no matter its resources. This is also why open standards-based technology is always more secure than those based on closed standards: A broad ecosystem is stronger than a single entity.

Of course, we haven't talked about the elephant in the living room: If Microsoft wants to compete for business in the State of Massachusetts, all it has to do is support the ODF standard in its Office suite. Of course, that's up to them—I'm sure there are complex technical issues that go into this decision—but most likely it's the business and licensing issues that trip up Microsoft. They would rather control the fate of their own proprietary formats.

As more end users rise up and seize control of the future of their data and applications, the technology industry (especially software companies) will have to do more than offer lip service to open standards. They will actually have to build businesses around them.

Jim Zemlin is executive director of the Free Standards Group. He previously served at Covalent Technologies and Corio.
e-Mail: jzemlin@freestandards.org
Website: www.freestandards.org

BY RAVEN ZACHARY

New Models of Support

Beginning with this edition of *Enterprise Open Source Journal*, The 451 Group, a technology analyst company, and I will be providing a regular column on the Commercial Adoption of Open Source (CAOS).

One of the issues on my mind recently is a topic many have found painfully uninteresting: the issue of software support. Within the context of open source, however, this is anything but dull.

The freedom of choice provided by the use of Open Source Software (OSS) brings with it a greater availability of support options. Customers with access to source code have support options that historically weren't available to them with proprietary software.

This public access to the source code provides an opportunity for the creation of expertise (and sustainability) outside the core development team, resulting in these new support options. For some customers, the flexibility in selecting a support model is enabling, while for others it has been a major barrier for open source adoption.

The issue for customers is that support has traditionally been included when selecting software. A single proprietary vendor has provided the software and related support services. In contrast, not every successful open source project is backed by an open source vendor; in fact, few are. Selecting a support model is now a decision that has to be made in the software evaluation process.

Before JBoss or Covalent provided support for Tomcat, I hired a member of the Tomcat development team to provide my previous employer with the expertise it needed to support a multi-million dollar e-commerce system. Hiring was just one aspect of building a support model for our business, motivated somewhat by a lack of vendor support for Tomcat at the time.

So, what's the preferred support model for open source customers? Obtain support directly from an OSS vendor? Rely on support from the community? Build internal expertise through training and hiring of expertise from outside? There's no single answer that meets the needs of all customers. Through the increased adoption of open source, there has been a fundamental shift of responsibility back to the customer when it comes to support.

While most open source vendors have pursued a mostly traditional support model to date, the nature of open source has created the opportunity to provide new models of support specifically tuned to the unique needs of the open source customer.

Support Trends

Here are some trends I see developing with open source support in the near future:

Greater demand for acquired talent: Core developers for successful open source projects will be in higher demand as the adoption of open source continues to accelerate. Demand is growing at a rate greater than the availability of talent. If you're planning to hire talent, this will become increasingly difficult.

The use of transition talent: If your organization is in the process of building internal open source expertise, the use of external open source talent ("consultants") is an effective way to provide temporary support during the transfer of these skills to internal resources.

Self-sufficient enterprises: More and more customers are building their own internal support organizations for OSS, with limited or no vendor dependency. This model won't be the predominant one, but we're already seeing this trend among some large enterprise customers with the resources to pursue this path.

Increased need for training: Customers see the value in having internal open source experts on staff, even if they rely on a vendor for support. Training companies are poised to profit from this demand, due to the constrained availability of acquirable talent I mentioned earlier.

Support aggregators: We're already seeing this trend with companies such as SpikeSource, OpenLogic, Covalent and SourceLabs, but I expect to see further movement in this area by companies that want to be the primary open source support vendor for customers struggling with building a mix-and-match support solution.

More "try before you buy": With more options available to customers, support providers will need to offer creative support options, including those catering to customers in the evaluation process. As an example, Laszlo Systems provides this service for OpenLaszlo, an open source framework for rich Internet applications.

Outright competition: There's no practical reason why open source vendors such as JBoss and MySQL haven't faced direct competition for support from third-party vendors. Once the market is large enough, there will be activity in this sector. The barriers to entry are low.

As the adoption of open source continues to increase, expect to see new and innovative models of support emerge. ●

Raven Zachary is a senior analyst and the open source practice head for The 451 Group, a technology analyst company. At The 451 Group, he's responsible for the Commercial Adoption of Open Source (CAOS) Research Service.
e-Mail: raven.zachary@the451group.com
Website: www.the451group.com

REALITIES ABOUT OPEN SOURCE

AN EVOLUTION RATHER THAN A REVOLUTION

BY MATHIEU POUJOL

Open source is a technical advance much like those that preceded it: it's changing some models, requiring players to adapt, and it's boosting, but not revolutionizing, the market. The client/server and PC wave have required some companies, such as IBM, Software AG and Fujitsu, to take a look at their businesses and make changes.

Open source must be included in a maturity acceleration phase in the IT market that has emerged around phenomena that affect three aspects of IT:

- **Telecommunications:** the Internet, decreasing prices and democratizing access to communication networks; the Internet is the catalyst for other phenomena
- **Services:** relocation, whether at an external or foreign provider
- **Software:** open source.

These three points accelerated during the "Internet bubble," which multiplied their effects on the market, particularly in terms of prices.

A Stimulating Effect on the Market

Open source has put into question numerous IT market segments by becoming a credible alternative to some oligopolies and monopolies. This has had several major effects:

- Open source is a powerful advocate for standardization, thanks to its openness. Standards are critical for prices to drop and information systems to become more open. Standards are the nemeses of monopolistic situations.
- Open source has offered solutions in market segments where competition had dried up. It's an ideal threat to make software suppliers' prices fall. Moreover, these offerings are jumpstarting the basic offerings. There's no longer a need to choose a product that's too expensive because it has too many functionalities, whereas the need is rather simple. There's Open Source Software (OSS), as well as commercial solutions that are aligned with the cost of open source.
- Open source has redefined and cleaned up the notions of criticality and commodities in IT:
  - Faced with OSS, "traditional" software suppliers are lowering the prices of their products and boosting their offerings by innovating and offering higher value-added and more reliable solutions.
  - Customers buy only what they need and don't pay for superfluous functionality. This is the "good enough" effect.
  - Open source, through its collaborative and communal aspect, has forced software designers out of their ivory towers and brought them closer to market expectations by boosting specific development. The result is innovation in information systems.
  - OSS is a great laboratory for testing IT concepts without having to bear a heavy financial burden.

Figure 1: Software Industry Supply Chain

- In the public sector (especially in France), open source can reduce production time for a solution, since in some cases it's not necessary to have public calls for tender. This is one of the main reasons for the success of open source in the public sector because the procurement processes for public markets are constraining.

Standardization, openness, accrued competition, decreased prices, a race to innovate, and a better correlation between needs and offerings. All IT players owe it to themselves to recognize the benefits of OSS for the whole industry.

New Models

OSS has enabled the emergence of a new value chain and new business models. Figure 1 shows the software industry supply chain and Figure 2 shows the open source supply chain. The value chain for open source is more complex, which can be a source of difficulty if its selection process isn't well-structured. The contractual relationship between the end client and the different participants can become problematic for those involved.

Important Role Changes

OSS also changes the relationship between software suppliers and IT services companies. To make up for weak revenues from software sales, software suppliers specialized in open source sell in volume and offer services. This is the main way for "commercial" software suppliers specialized in open source to ensure their long-term durability.

To make up for the lack of functionality in products and the absence of strong maintenance capabilities by OSS suppliers, IT services companies step in and move toward the software supplier business. They even take on contractual obligations. IT services companies regain the added value that they'd lost to software suppliers over time to the software editors that have been packaging the different layers of the information system.

This rise in the added value of services in the IT market has been a perpetual and weighty trend. It's forcing software to become more complex and to move further toward processes. Many software suppliers are increasingly betting on services related to their software as a main revenue model.

In addition, IT services companies, armed with their industry knowledge, appear to be incorporating more and more packaged software components in their solutions. This software may be built with OSS stacks or specific business components that the IT services companies developed. The arrival of Service-Oriented Architecture (SOA), based largely on software development oriented toward components, only reinforces this trend; IT services companies build software and software components around their internal frameworks, then integrate them in the solutions they offer their customers. As in the past, IT services companies are becoming software suppliers again due to the influence of OSS and SOA.

Technical Neutrality

OSS, when well-chosen and well-used, lets companies avoid certain technical constraints imposed by traditional software companies, including vendor dependence and restrictive, expensive licensing systems. However, this neutrality also can be achieved through "commercial" software insofar as they respect official and market standards.

Open source can lead to strong constraints in a company that can find itself a prisoner of its choices, since (as for all software needs) specific development around a platform is risky. There's often more specific development on open source than on proprietary software, since the needs not covered are generally greater than those for a packaged solution; one can also run the risk of returning to systems that are too specific, where knowledge of the system will rest on the IT services company or on individuals in the company. This is risky, too, particularly if there's not an extremely good project management framework. It's an approach that should be justified only for highly specific needs.

When it's not justified, companies must avoid going back to non-standard systems. Technical neutrality is a key point when investing in IT. One of the principal dangers of OSS is to favor approaches that are too specific; on the other hand, one of its main advantages is technical independence. The solution for all software investments, whether "open" or not, is to follow market standards.

The Return of Specific Development?

Open source has boosted specific development. It has provided developers with a whole range of technologies that are highly adjustable and adaptable to user needs. Open source is an innovation driver

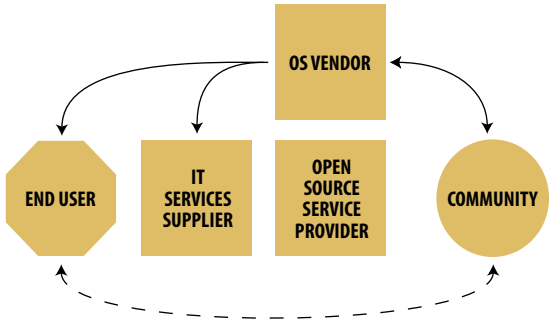


Figure 2: Open Source Supply Chain

in software technology that makes some specific development more affordable. Moreover, "traditional" software suppliers are licensing aging or non-marketed products as OSS, thus increasing the number of components available to create specific software from common and free software. These open systems also rely on an ecosystem of open source and proprietary software (such as Zend Studio) of higher and higher performance, such as Linux, Apache, MySQL, and PHP (LAMP), all widely used for Websites.

Specific development is increasingly framed by tools (e.g., project management, governance) and methodologies (e.g., Six Sigma).

It's possible to make increasingly inexpensive custom systems by relying on open source technologies and existing development. However, this approach must occur within a strong framework in terms of project management and must be limited to either common or highly specific needs. This shouldn't be a step backward toward completely custom systems; the trend is toward the packaging of IT systems.

It's important to conform to standards and to evaluate investment over the total lifetime of the solution and its mutations (often five or more years in big projects). It could be better to invest in a traditional software supplier's solution if the products are easier to manage and to maintain and the costs are easier to define. This is particularly true for the company's most structured software, such as a platform or an application foundation such as Enterprise Application Integration (EAI), application servers, transactional monitors, databases, administration platforms, etc.

Consider the model that emerges as a hybrid model between the specific and software packages, a model that's similar to a game of Legos, where sometimes specific stacks are built among themselves to create a coherent system. Some such stacks are created from scratch. Open source acts as a strong driver behind this evolution by supplying stacks and components without the need to pay for licenses. This trend continues to be favorable for IT services providers. A company must evaluate whether specific development will create more value than if it were a software package.

This depends on the:

- Company's economic sector
- Company's existing IT
- Role of IT in creating added value
- Company's internal IT capacities and IT project management capabilities.

The Future Model

The future model is an industrialized specific development from standardized components. Within the industry, this model is similar to the Dell model: specific that's less expensive than packaged, relying in large part on basic normalized, standardized components.

Applied to software, this model allows for specific development that's less onerous than a standard solution. Thus, all Enterprise Resource Planning (ERP) suppliers are now looking to open their products and to "deintegrate" them by placing them on integration platforms, the packaged suites on which the enterprise applications are built according to n-tier architectures. These platforms carry the SOA model. An integration platform gathers an application server, a portal, an integration server, a development environment, and a Business Process Outsourcing (BPO) tool on one platform.

Open source, when it's normalized, is an important aspect of this componentization of IT architectures by allowing easier and legalized access to either common or highly specific software resources. It's important to evaluate what must be specific in the creation of value from what can be packaged. The specific is more dependent on open source.

This renewal of specific, structured, industrialized, agile, and flexible development requires setting up an SOA. Open source is still too often perceived as a response to a specific, limited need. It's already used to resolve tactical issues but companies have everything to gain if they see them as part of a long-term policy, in a strategic approach within their future SOAs. Open source is a factor in reducing costs and also a tool to create added value.

For a company that's conscientious about creating a competitive advantage using its IT systems, SOAs are essential. It's better to do specific development and integrate commercial and OSS packages within an architecture that's capable of receiving and managing them than to do specific development on an integrated software package or specific development starting from zero. The SOA is the clutch between a company's functional needs and its IT resources.

The model in Figure 3 requires strong project management capabilities and a high level of technical maturity. The eternal choice in IT between "build" (specific) information systems and "buy" (packages) information systems remains a problem. The pendulum between software packages and specific development appears to again be swinging in favor of the latter, thanks to open source and SOAs; software packages will be increasingly seen as components that are increasingly standardized and packaged. To design such architectures, it's imperative to invest strongly on two levels:

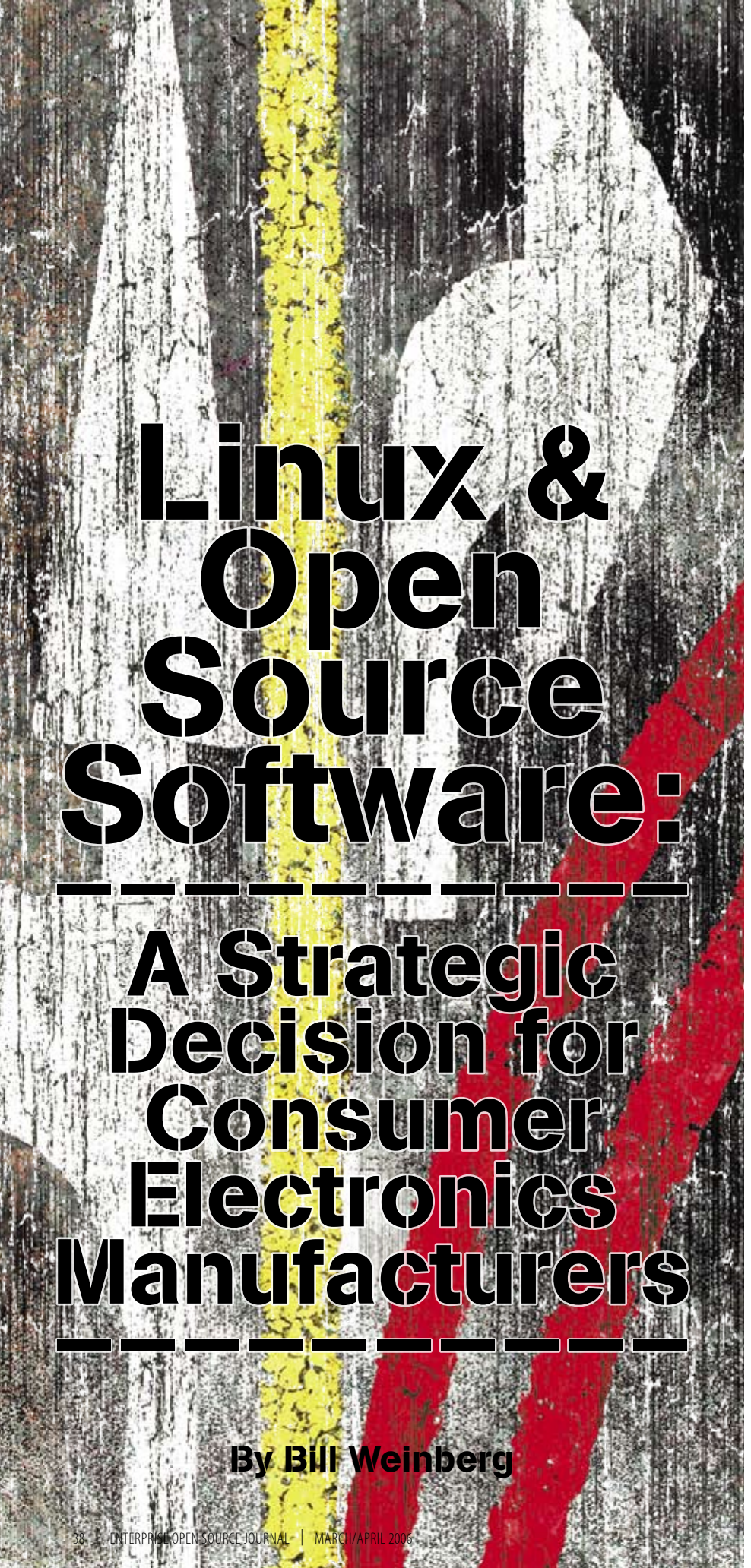
- Upstream consulting to establish a coherent city planning process of the information system within companies' competitive environments, since two-thirds of IT project failures are related to upstream phases
- The integration platform that will serve as the SOA's engine. This is the "operating system" of Internet architectures.

Solid construction requires a good foundation. To maximize open source investments in an IT architecture, it's better to evolve toward a solid SOA. Open source must be considered within an overall city planning process where the OSS can be integrated within heterogenous software based upon its capacities.

This IT model is effective, well-adapted, agile, highly innovative (and therefore a creator of value), and capable of differentiating the company from its competition. ●

Figure 3: SOA—The Clutch Between Functional Needs and IT Resources

Mathieu Poujol is a consultant with Pierre Audoin Consultants (PAC), specializing in Systems Infrastructure Software. PAC advises IT companies on achieving domestic and international growth objectives in Europe and the U.S. through the planning, development, implementation, and ongoing support of successful growth strategies.
Voice: +33 (0)1 56 56 74 17
e-Mail: m.poujol@pac-online.com
Website: www.pac-online.com



Linux & Open Source Software: A Strategic Decision for Consumer Electronics Manufacturers

By Bill Weinberg

Less than a decade ago, hardware content dominated all categories of embedded applications; device Original Equipment Manufacturers (OEMs) were first and foremost hardware manufacturers. OEMs afforded embedded operating systems and the programs hosted on them the same status and importance as capacitors and diodes that shared board-space and Bill of Material (BOM) cost with software-centric CPUs and Dynamic Random Access Memory (DRAM).

Today, by all measures, software content is king, eclipsing hardware in importance, cost and most important, perceived added value. Software, not the hardware where it runs, lets device manufacturers differentiate their wares and survive in fiercely competitive markets.

Forward-looking Consumer Electronics (CE) manufacturers understand that software lets them build intelligent devices. For these firms, choice of software platform is increasingly a strategic decision, going beyond the bits, bytes, and specs of kernels that once dominated platform debates. This article describes this new software-centric world of CE and how leading CE manufacturers, start-ups, and upstarts view Linux and Open Source Software (OSS). In particular, it recounts who is using Linux and in what types of applications, the driving forces behind its adoption, and how internal and external software ecosystems evolve around Linux-based device platforms.

Adoption Trends

Analyst firm Venture Development Corp. (VDC) reports that in 2005, Linux-based OSS garnered 25 percent of new 32- and 64-bit design wins, with 29 percent of developers planning to use Linux in their next project (VDC data from a May 4, 2005 Ziff Davis eSeminar).

This leadership position places Linux well ahead of traditional embedded (Real-Time [RT]) OS platforms such as Wind River VxWorks (12 percent) and Microsoft embedded platforms (Window CE, XP embedded et al.). In the Linux embedded design space, VDC also cites the leading design domain to be CE, at 35 percent, followed by communications (30 percent), and industrial control (11 percent).

CE is actually a poor term to describe today's mass market of intelligent devices. To most readers, CE implies first in-home entertainment appliances such as televi-

MOBILE & WIRELESS	TV & HOME ENTERTAINMENT	AUTOMOTIVE TELEMATICS & IN-CAR ENTERTAINMENT
• Mobile Phones	• Digital / HDTV	• Navigation Systems
• Wireless PDAs	• PVR / DVR	• Vehicle Management
• Portable Media Players	• Set Top Box	• Digital Radio
• Portable Game Consoles	• Digital Audio Receivers	• Digital Media Players
• Intelligent Remote Controls	• Musical Instruments	• Hand-free Mobile Phones
• Digital Still and Video Cameras	• Karaoke	• Wireless Data and Media Sharing
	• Game Consoles	

HOME NETWORKING & CONTROL	SMALL OFFICE & IMAGING	
• Home Gateway	• Laser and Inkjet printers	• Routers, Firewalls, VPN
• Broadband Access	• Fax & Scanners	• IP Telephony Clients
• Home Automation	• Intelligent Copiers	• Audio & Video Conferencing
• Security & Monitoring	• Multi-function Peripherals	• PBX & Voicemail
• Domestic Robotics	• Network Printers	

Figure 1: Categories and Types of CE Devices Deploying Linux and Open Source Software

sions, video equipment, and video games. Next to come to mind will be in-hand applications such as media players and cell phones. However, the field is both broader and deeper: In-car is an important CE category with navigation systems, graphical dashboards, and digital entertainment systems; with nearly ubiquitous broadband connections in most major markets, in-home has expanded to include low-cost networking, security, control, and Small Office, Home Office (SOHO) applications.

Figure 1 summarizes the five segments of the CE market, with Linux and OSS participating in all the segments and applications listed.

In-home, in-hand, in-car and elsewhere, Linux is attracting a who's who of CE manufacturers. The lion's share of CE design and manufacturing occurs in Asia: Tigers in Japan, China, Korea and Taiwan have all invested heavily in conversion from purely proprietary software to a mix that includes Linux and open source (as well as in-house and commercial proprietary software). The same goes for European and American manufacturers, especially in automotive and mobile telephony.

Figure 2 lists companies with announced Linux strategies, many with Linux-based product lines shipping since 2002.

Driving Down Cost

The initial, tactical attraction to Free and Open Source Software (FOSS) is cost reduction. Immediate savings from FOSS arise from software acquisition—free access to source code and lower (or even zero) costs for development kits. For high-volume CE devices, freedom from run-time licenses or royalties for significant parts of the platform help OEMs improve bottom lines. And, after the current project ships, device OEMs have more leverage in negotiating toolkit licensing costs (if any) for follow-on applications, further reducing development and deployment costs.

FOSS isn't a panacea. Manufacturers who decide to roll their own Linux and FOSS-based stacks can end up spending significant engineering monies if they lack sufficient expertise. When OEMs choose commercial FOSS-based solutions, licensing costs also can approach the levels of legacy proprietary solutions. While FOSS deployment may be royalty-free, OEMs can still find themselves owing per-unit fees for remaining proprietary components, or even for FOSS components as part of unit-based support contracts.

Ultimately, FOSS' biggest impact on total costs comes through giving device OEMs more choice among technologies, vendors, and solutions.

Platform Consolidation

Device OEMs face a bewildering diversity of processor types, system architectures, development tools, programming languages, and embedded operating systems. On the hardware side, CE applications sport processors that include versions of ARM, DragonBall, M68000, MIPS, PowerPC, SuperH, and other CPUs.

Device OEM product teams are also organized along hardware-centric lines. In many CE companies, each product or line builds on unique hardware and deploys unique OS, middleware and applications, eschewing the obvious economies of scale those same companies leverage in their manufacturing. Platform fragmentation in companies and across the industry is further exacerbated by acquisitions and mergers that bring divergent hardware and software into already diverse engineering environments. Examples include Nokia's takeover of Brazil-based Gradiente mobile, Cisco's purchase of Linksys, and BenQ's acquisition of Siemens' handset division in 2005.

Starting in the late '90s, leading CE suppliers (such as Sony and Panasonic) began to search for a strategic platform. The Japanese embedded market then was dominated by derivatives of the iTRON embedded OS, which, while ubiquitous, provided a limited set of services and functionality. Taiwanese, Korean and European manufactures found themselves in similar positions with commercial Real-Time Operating Systems (RTOSes) such as VRTX, pSOS and VxWorks. Coincidentally, in the same timeframe, enterprise IT giants, including IBM, HP, Fujitsu and others, began looking for a platform to unify their diverse system architectures on the hardware side, and multiple flavors of Unix and OSes such as CMS and OS/3xx on the software front.

In 1999 and 2000, after a decade of incubation, Linux sprang onto the scene. Through investment by the aforementioned manufacturers, by Linux-based

Acer	Garmin	NEC	Sharp
BenQ	Haier	Nokia	Siemens
BMW	Hewlett Packard	PalmSource	Sony
Canon	Kenwood	Philips	Sylvania
Casio	LG	Pioneer	3COM
D-Link	Linksys	Roomba	Tivo
Epson	Matsushita	Royal	Toshiba
Ericsson	(Panasonic)	Samsung	Volvo
	Mitsubishi	Sanyo	Yamaha
	Motorola		

Figure 2: CE Manufacturers With Announced Linux Strategies and Active Linux-Based Products

platform suppliers such as Red Hat, SuSE and MontaVista, and by silicon suppliers such as Intel, AMD, Motorola (now FreeScale), Texas Instruments and ARM, Linux quickly acquired the hardware support and capabilities to be that strategic platform, in both enterprise and embedded applications.

As a strategic platform for embedded computing, Linux makes sense. It lets device OEMs manage increasingly large, complex application loads, using the same capabilities that make Linux such a good enterprise OS. It helps those same companies streamline development teams and processes, optimizing training, development, and maintenance investments by providing a common platform that crosses CPU hardware and application boundaries, reducing redundancy of effort while enabling OEMs finally to enjoy the promised benefits of software reuse across product lines. Moreover, since Linux isn't the Intellectual Property (IP) of a single company, no single vendor must recoup the staggering R&D cost of writing kernel and user applications from scratch, as many device OEMs had to do for their legacy embedded OSes and stacks.

Open Platform

While device OEMs began by focusing on consolidating the hardware and software hidden inside their wares, the CE marketplace was shifting its emphasis from standalone devices to connected platforms for service delivery. Today, the perceived value of the application lies in its ability to deliver entertainment, voice, and other connectivity services.

Just as device OEMs product designs lived inside silos of hardware and software constraints, service delivery systems such as cable TV, phone service and Internet access depended on technologies and sales channels that were inextricably bound to the carriers and operators that provided them. Starting five years ago, attempts at defining common and semi-open standardized delivery platforms looked to Java as the "can opener" to cross over carrier and operator boundaries. For in-home entertainment, that meant Java-based Multimedia Home Platform (MHP) and Open Cable Application Platform (OCAP) middleware in cable boxes and television sets; for mobile telephony, it implied Mid-P Java profiles and also BREW Application Program Interfaces (APIs) deployed on cell phones.

While these and other software platforms offer common APIs and the promise of interoperability, they also raise deployment costs, and frequently fail to deliver on performance and finer-grained interoperation of applications, displays and multi-media. An extreme example comes from the gaming software segment—for the mobile handset market alone, game Independent Software Vendors (ISVs) report having to re-target their products dozens of times for variations in phone OSes and middleware complements: One ISV cited the need to configure their most popular offering in more than 100 device-specific versions.

Linux and other FOSS carry multi-faceted promise as a platform or set of solutions for open, standards-based CE service delivery. First, unlike legacy embedded OSes, competing platform offerings from Microsoft, Symbian and others, and also unlike Java, Linux provides a high-performance embedded OS with truly standard APIs and open standard implementations of networking protocols. By leveraging its desktop and enterprise roots, Linux offers device OEMs a more robust, secure OS, with support for dozens of legacy CPUs and next-generation, multi-core processors.

A good example of the value of Linux openness and crossover from enterprise deployment is routing. While stand-alone routers don't fall into most definitions of CE, many consumer devices need enterprise-class routing and security in a small package. For example, while the main function of cell phones and set-top boxes is to deliver voice and video, those same devices also boast WiFi, Universal Serial Bus (USB), Ethernet, IrDA (Infra Red Device Association), FireWire, and other interfaces with the need for seamless handoff and transfer of multi-protocol datastreams and sessions.

CE Software Ecosystems

With a shift in perceived value-added from hardware to software comes renewed requirements for integration of in-house and Commercial Off-the-Shelf (COTS) software. The estimated annual doubling of the number of lines of code in CE devices has outstripped the ability of even large device OEMs to maintain their own OS and software stacks, let alone develop them from scratch. Instead, CE manufacturers find themselves straddling two developer communities: a pre-platform software ecosystem that supplies the technologies and software components that let OEMs develop and deliver their products to market; and a post platform community that creates after-market applications and partners with service providers to deliver those applications and to support incremental services offerings on the devices.

Today, the emphasis for embedded Linux lies on the pre-platform side. Dozens of traditional and newly minted software suppliers, hundreds of open source projects, and thousands of individual open source developers contribute to the Linux kernel and key projects that sustain ongoing innovation in CE devices. These players come together through organizations such as OSDL's Mobile Linux Initiative (MLI), and the Consumer Electronics Linux Forum (CELF). This ecosystem effectively outsources the massive task of creating and integrating the base of the software stack (below and at the value line). Device OEMs add differentiating value above the line with industrial design, usability, manufacturing capability, distribution channels, marketing and branding, product support and partnering with service and content providers.

The post-platform ecosystem for Linux and for all device platforms is still coalescing. Indeed, post-platform marketplaces are usually product-specific (as was the ecosystem around PalmOS devices or more recently around TiVo) and ephemeral, coming and going with the fortunes of the device families at their centers. The formation of a Linux-based, post-platform ecosystem will hinge on several factors: first, the willingness of carriers and operators to open their networks; and second, the ability of Linux-based platforms to strike a balance between broad interoperability among devices and device types, while providing a sufficiently robust and segment-specific capability set to meet the needs of a given device. That is, drawing the value line both to support rapid device development and still leaving room for differentiation. ●

Bill Weinberg brings more than 18 years of open systems, embedded, and other IT experience to his role as open source architecture specialist and Linux evangelist at the Open Source Development Labs, where he participates in OSDL initiatives for Carrier-Grade, Data Center, and Desktop Linux.
e-Mail: bweinberg@osdl.org

BY LARRY SMITH

The Open Systems Scorecard: Part II

Some time ago, we considered the open systems scorecard, looking into the ancestry of Linux and the Berkeley Software Distributions (BSDs). Somewhat conspicuous by its absence is a relative newcomer to the open source scene, but it's no spring chicken—it's a powerful and mature operating system. In this issue, we examine what might be considered the grand old patriarch of the clan: Solaris.

To find out where Solaris began and where it wound up, we need to look at the evolution of Unix. As we saw in Part I (January/February *EOSJ*), Unix has had, nearly from the beginning, two different personalities: one was academic, personified in the BSDs, which eventually shook off AT&T and evolved into today's trio of BSD operating systems. But here, we're interested in the road not taken—not taken by BSD, anyway. This was the code base then known as AT&T's commercialized version of the genuine, official, accept-no-substitutes, certified Unix.

So, this is the real McCoy, the bearer of the Unix registered trademark. This Operating System (OS) has gone through several revisions over time (including picking up quite a number of features pioneered by its academic sibling BSD), evolving into the more-or-less "final" product dubbed "System V"—SysV, or SVR4, to insiders. It was this version AT&T eventually sold off, and it was this version that meandered through several owners and a variety of not-very-obvious transactions to become what it is today: the SCO and Novell football. It's a sad end for a great lineage. But I'm getting ahead of the story.

At this point, SysV was still a viable product, the apple of AT&T's marketing eye. SysV was walking through the woods one day, minding its own business, when it ran smack dab into a cousin, a BSD, no less. But not just another BSD—it was the BSD known as SunOS.

Sun was one of a handful of companies responsible for creating the modern workstation. They concentrated on hardware, but turned to BSD and its quite lenient license for their OS. The BSD license was designed to encourage companies to adopt it and take it "in-house," and this is what Sun did to produce the new, non-open source "SunOS." Sun sold this OS starting in 1982 and carried it forward into the late '80s.

So, AT&T's own SysV Unix met Sun's SunOS and fell in love. Well, maybe not love exactly—but the two companies decided to partner, and features from each OS began to cross-pollinate the other. Later on came the bitter divorce, leaving AT&T's official Unix to go on to face its horrible fate, and Sun with the OS now known as Solaris. Solaris is the latest incarnation of SunOS—with BSD and SysV extensions—coupled with a windowing system and GUI.

Solaris went on to become Sun's powerful and reliable workhorse, and it made a great many friends in the corporate world. It has always run on x86 machines, but is best-known as the OS of choice for Sparcs, both 32- and 64-bit.

Despite the fact that BSD and Linux can both run on Sparcs, it must be admitted that neither can really get as much sheer grunt from the architecture as Solaris can. Its biggest claim to fame is its fine-grained thread-locking, allowing it to scale gracefully and nearly linearly to 64 processors.

Solaris, which is still "real" Unix, though the trademark no longer applies—has returned to its open source roots.

Neither Linux nor the BSDs care to push their architecture in that direction. The more processors you can scale to, the more spinlocks you need in the kernel, and therefore the poorer the performance on fewer processors. Linux and the BSDs prefer to switch to clustering to deal with huge numbers of processors. They do scale, and they do work—but they are just not as beautifully optimized for large numbers of processors in a single box. If you have a lot of processors and you prefer not to use clustering solutions because of the overhead, well, Solaris is your best friend.

Just a few months ago, Sun finally did something it has been thinking about for some time—it released Solaris as an open source operating system under the OSI-approved Sun Common Development and Distribution License. This, as Yogi Berra once said, is déjá vu all over again—Solaris, which is still "real" Unix, though the trademark no longer applies—has returned to its open source roots. And I say, "Welcome back!" ●

Larry Smith started at Digital Equipment in 1978, and has accumulated 27 years of experience in software engineering. He is presently a consultant at Wild Open Source, specializing in quality issues and user interface design.
e-Mail: larry@wildopensource.com

BY HOWARD FOSDICK

What's With Object Rexx?

Previously, I've discussed how IT organizations are capitalizing on open source scripting languages. These languages offer the flexibility and productivity associated with interpreters while fitting the new open source, Web-based paradigm for software support and community. Having previously looked at the KornShell and Perl, we'll wrap up our tour with Open Object Rexx.

Open Object Rexx, or ooRexx as it's called, is an interesting creature that's received a lot of attention. Shortly after its introduction in April 2005, it garnered 24,000 downloads on the SourceForge Website—enough to place it in the top-3 percent of all downloads last spring. What's up with that?

ooRexx was developed at IBM 10 years ago as a superset of its Rexx scripting language. Procedural Rexx—now called "classic Rexx"—runs on any imaginable platform, and predominates on several. The language gained an ANSI standard in 1996. If you were to summarize Rexx in a phrase, "easy to code yet powerful" would be it. Rexx bases its power in clean, simple syntax. This contrasts to languages that evolved out of the Unix tradition, such as the KornShell or Perl.

ooRexx adds Object-Oriented (OO) features to classic Rexx for OO scripting. These include classes, messaging, single and multiple inheritances, encapsulation and data hiding, operator overloading and polymorphism, and a large class library.

IBM open-sourced ooRexx in December 2004. The Rexx Language Association took over development and support. The "RexxLA" has since repackaged the product for Linux, Windows, and Unix. You can find the language and its documentation at the new ooRexx Website at www.oorexx.org.

The key to ooRexx's success is that it extends classic Rexx into OO programming while maintaining 100 percent compatibility with standard Rexx. Any classic Rexx script runs, without change, under ooRexx. This yields portable scripts and transferable staff skills.

Most mainframe and former OS/2 and Amiga developers know classic Rexx. With ooRexx, they can transition to OO scripting at their own pace, write traditional procedural scripts, and then add a few OO features as desired, switching completely to the OO paradigm over time. Any programming problem that's best addressed procedurally can still be coded procedurally.

Open Object Rexx applies Rexx's ease of use to OO scripting. This offers all the benefits claimed for OO programming:

- Simplified design through modeling with objects
- Greater code reuse
- Rapid prototyping
- Higher quality of proven components
- Reduced maintenance
- Cost-savings
- Increased adaptability and scalability.

ooRexx leverages the many free Rexx tools available on the Web. The language is extensible in that you code and use classes and functions in external packages just like those included in the core language. ooRexx comes complete with several tools, including:

- **Rexx Interpreter:** The free, open source Rexx interpreter
- **Rexx Tokenizer:** Tokenizes a script for faster execution and hides source code
- **RexxUtil:** Extended functions for operating-system independent programming
- **RxSock:** Extras for working with TCP/IP sockets
- **RxMath:** Add-ins for transcendental math
- **RxRegExp:** Supports regular expressions
- **Excellent documentation:** Highly readable introductory and reference manuals.

On the Windows platform, ooRexx integrates with key operating system features, including ActiveX and Object Linking and Embedding (OLE), Windows Script Host (WSH), Active Directory Services Interfaces (ADSI), and Windows Management Instrumentation (WMI). WSH support allows you to write Windows system administration scripts with ooRexx. Microsoft posts more than 120 examples of ooRexx sysadm scripts for free download.

ooRexx combines easy OO scripting with real power. Find further information at the Rexx Info Website at www.RexxInfo.org and at the ooRexx Website at www.oorexx.org. ●

Howard Fosdick is the author of the *Rexx Programmer's Reference*, a new book that covers scripting for Windows, Linux, mainframes, handhelds, Open Object Rexx, and the major Rexx tools and interfaces. Find it at www.amazon.com/rexx.
e-Mail: hfosdick@compuserve.com

Open Source Content Management Systems Deliver a Low TCO

BY VASUKI KASTURI

A Content Management System (CMS) supports the creation, management, distribution, publishing, and discovery of corporate information. A CMS solution needs to empower casual users, knowledge or content owners, power users, and so on. Ideally, an organization should be spending 75 percent of its content-related budget on content creation and 25 percent on content management.

Open Source Software (OSS) has become increasingly mainstream. Open source CMS solutions have matured enough to be considered alongside commercial alternatives.

Implementing a CMS can be among the largest IT projects an organization undertakes, but by nature, offers a low Total Cost of Ownership (TCO). Making a straight comparison between any two products is challenging and any client exploring this space should be prepared to spend considerable time in deep investigation. This article seeks to identify considerations that can assist in due diligence.

Overview

Today's competitive landscape has accelerated the speed of business, forcing enterprises to operate in real-time and respond instantly to changing conditions. The best companies are responding by enabling access to information as soon as it changes. Today, success is increasingly defined by how efficiently you capture, create, manage, and deliver information. Information is at the core of your business, and delivering the right information, at the right time, determines the effectiveness of your communication strategy.

The challenge of managing an enterprise Web presence has never been greater. Web properties are critical assets; they're driven by various combinations of CMSes and portal packages. Some are custom-built, others are commercial packages, and still others are built using open source solutions. All the properties differ in maturity but share the same need for relevant, timely, frequent content changes.

Traditional processes of maintaining Web properties have been inefficient, expensive, and error-prone. Smaller enterprises, looking for a corporate presence, have their Web properties developed and maintained by dedicated Web teams, a scenario that has lead to a single point of failure. Large enterprises tend to transform their Web properties into strategic assets. In either case, ineffective Web content management can significantly undermine corporate messaging, decrease sales, increase staffing requirements, and raise costs and risks. Without automated workflow, the Web content quality suffers because a formal approval mechanism is lacking. Organizations that lack a content management strategy or have an inefficient one fail to respond in real-time (see Figure 1).

Business Drivers

CMSes, a combination of large database, file system and other related software modules used to store and later retrieve huge amounts of data (see Figure 2), were created to address the problem of managing more and more content of a variety of types, as fast as possible. These challenges required automation and management. The business drivers for having a CMS are:

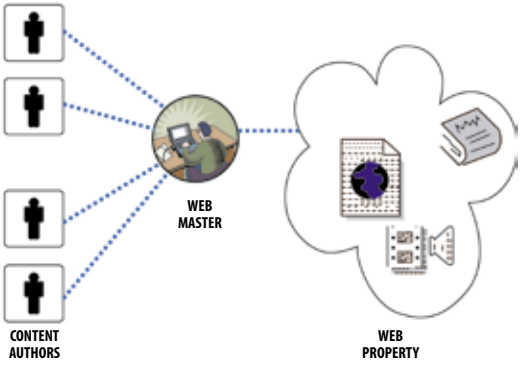


Figure 1: Without a CMS

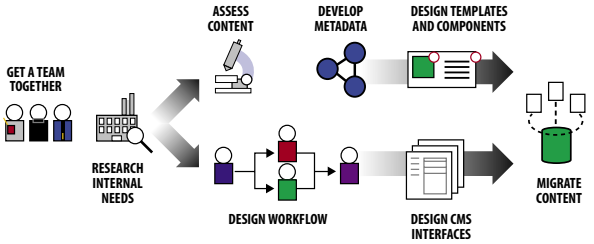


Figure 2: Lifecycle of Content Management

- Diluted brand presence
- Costly Web development staff
- Outdated or erroneous content
- Poorly managed or non-existent workflows
- Lack of version control or backups
- Redundancy of effort.

A CMS supports the creation, management, distribution, publishing, and discovery of corporate information. It covers the complete content lifecycle, from providing simple tools to create the content to publishing and archiving. It helps manage the structure of the Web property, the appearance of the published content, and the navigation provided to users. Actually, CMSes are broader than this.

Business Benefits

A wide range of benefits can accrue from implementing a CMS:

- Streamlined authoring process
- Faster turnaround time for new pages and changes
- Greater consistency
- Improved site navigation
- Increased site flexibility
- Support for decentralized authoring
- Increased security
- Reduced duplication of information
- Greater capacity for growth
- Reduced site maintenance costs.

Moreover, the greatest benefit is better support for your goals and strategies. By having a consistent, clear message, a CMS assists in communicating with the public. By providing an intuitive system that enables quick content creation, a CMS can bring more visitors to the Website, improve customer satisfaction, and enhance sales prospects.

Enterprise Software Model

Most organizations have invested in a robust infrastructure. Unfortunately, shrinking budgets have left IT organizations lean and more concerned with essential services than large, new projects. An "enterprise-wide" CMS solution costs many thousands of dollars in addition to infrastructure costs.

Commercial CMS solutions are often marketed as shrink-wrapped, out-of-the-box solutions, but are usually "consulting ware"—whose installation, configuration, customization, and maintenance need numerous consultants and cost the enterprise thousands of dollars. Enterprise software today has several inherent problems, including:

- Longer sales cycle so expenses are added to the price
- Prohibitive costs
- Inaccessibility to small and medium-size businesses
- Disconnect between license cost and manufacturing cost
- Long customization cycles
- Inflexibility with the closed architecture.

Loss of license revenue has forced these commercial vendors to increase other cost components such as maintenance, customization, and support.

Open Source Promise

OSS has gained broad acceptance with the growth of the Internet and popularity of Apache and Linux. Organizations are opting for clusters of "cheaper" Linux servers instead of monolithic, expensive, proprietary boxes. The field of CMSes has seen strong growth in OSS solutions, perhaps in response to the high prices of commercial CMSes.

Benefits of OSS CMSes include:

- Low cost
- Ease of customization
- Platform independence
- No "lock-in"
- Better integration support
- Community support
- Documented systems.

OSS CMSes won't replace commercial offerings, but they offer a viable CMS alternative for many businesses. Solutions such as Zope CMF and Typo3 are almost "enterprise-ready." OSS solutions should be evaluated side by side with commercial CMSes in the demonstration process. OSS solutions may prove to be the best solution for some organizations, but whether it's open source or commercial, a CMS solution should be evaluated based on business requirements.

Selecting a CMS Solution

Choosing the right CMS is challenging. The CMS solution should be justified as an ongoing expenditure. There's no "one-size-fits-all" solution, because no two organizations have the same requirements. Neither does there exist an "out-of-the-box" solution: Some customization is usually required.

Three steps can ease the selection process:

1. Specify goals

System usage
- Web usage statistics
- Search engine usage
- Messages sent/posted
- Other knowledge creation measures
- Knowledge use

Number of users

Information quality
- User rankings
- Expert evaluation
- Edits required
- Usability testing
- Track-back links

User feedback

Maintenance costs
- Reduced staff
- Reduced capital

Staff efficiency
Printing costs
Distributed authoring
Process efficiency; reduced time
Transaction costs

Figure 3: Implementation Metrics

2. Identify requirements
3. Evaluate products.

Goals

Begin by determining the goals the CMS implementation will achieve. These should reflect long-term business strategies and directions; they should be well-understood and agreed to by all stakeholders before you begin the requirements-gathering process. For example, a CMS to manage a large corporate Website for a retail business might list these goals:

- Increase Website audience
- Reduce customer support costs
- Reduce duplication of information
- Increase flexibility of the site
- Improve the customer experience.

An intranet project, however, would list different goals:

- Improve staff efficiency
- Reduce publishing costs
- Reduce duplication of information
- Capture business knowledge
- Support knowledge discovery.

Use metrics to make your goals tangible and measurable.

Requirements

After you identify goals, you can begin the requirements process. There's no single best list of requirements because every organization has unique needs. Involve all your stakeholders in the requirements process, particularly if you're purchasing an enterprisewide CMS. Map each requirement to one (or more) goal(s); the requirements specify the "what," while the goals are the "why." Together, they form an integrated strategy.

For example, the goal of "reduce duplication of information" could lead to these requirements:

- Manage the Internet and intranet from the same system
- Integrate the CMS with existing systems
- Activate functions from a single source.

Because the requirements list can grow quite large, you should group the items into categories. Classifications that cover the entire CMS lifecycle could be:

- Content creation
- Content management
- Publishing
- Presentation
- Contract and business
- Evaluate vendor products.

After you identify requirements, use them for vendor selection. Ask vendors to provide detailed descriptions of how their system will meet your requirements. Using this approach helps ensure vendor accountability for any promises or commitments they make regarding their CMS. Ensure that vendor demonstrations are more than a sales pitch. Vendors must show how their product will meet your needs. The best way to achieve this is to develop scenarios—descriptions of common or important tasks that will be performed using the CMS. By presenting these in a "narrative" form, considerable scope can be covered in a relatively brief description.

Assess the implementation methodology, paying particular attention to the nontechnical aspects (such as training, change management, usability, and information architecture).

The TCO, not just the initial purchase price, may be an important consideration. TCO may include:

- Amount of customization required
- Technical skills and knowledge required by internal staff
- Degree of ongoing reliance on the vendor
- Licensing models and fees
- Third-party products required
- IT infrastructure required.

Whatever evaluation processes are followed, you must choose a single successful vendor. To do this impartially, create a scoring system. Score each of the requirements mapped to the business goals. Determine this before you contact the vendors, and incorporate the results of any demonstrations. Using a formal scoring system helps eliminate the potential for accusations of bias or corruption (for a sample copy, please visit www.cignex.com/site/library/datasheet).

Implementation

When the complex process of selecting a CMS vendor is complete, the success of the CMS project largely depends on how it's implemented and used. Implementing a CMS presents numerous challenges, including:

- Variables such as usability, architecture, and change management
- Staff participation
- Integrating with (or modifying) many business processes
- Implementing the CMS as part of a broader information or knowledge strategy
- Implementing a relatively new, immature product within the administration of the agency
- Interrelating to other information systems such as document and records management
- Ensuring long-term viability of the system and supporting processes.

These challenges introduce risks the implementation team must carefully manage. Prepare a checklist beforehand to identify key tasks that need to be addressed to mitigate the risks (visit www.cignex.com/site/library/datasheet for a sample checklist).

Measuring Value

If possible, specify metrics for measuring the success of each of the goals. Metrics provide a basis for calculating ROI and tracking the health of a CMS; they also help identify problems quickly enough for them to be effectively addressed.

With metrics still an area of ongoing research, you need to determine the best measures to use in your project. The table in Figure 3 lists some implementation metrics that can be tracked. You also can create metrics for corporate, customer service, etc. The metric for "improved customer experience" might be to "increase customer satisfaction with the Website to 90 percent, as measured by customer survey." Alternatively, "reduce support costs" could be measured by analyzing both Website usage and call center volumes. To help ensure effective metrics, measure them before the project starts to provide a baseline for comparison.

TCO

CMS has developed into a distinct, enterprise-level discipline. But with shrinking IT budgets, can content management continue to be justified as an ongoing expenditure? How can organizations know content management is delivering a real ROI?

Let's assume CMS can be justified on the basis of various intangibles. Organizations looking to measure the ROI must evaluate the cost of not having a system in place. Some difficult questions need to be asked:

	Year 1	Year 2	Year 3	Year 4	Year 5
Servers	50				
OS/database	25	5	5	5	5
CMS software	600–2,000				
Product support	125–370	125–370	125–370	125–370	125–370
Implementation	300				
Customization	200	50	50	50	50
Training	50				
Application support	25	25	25	25	25
TOTAL	**$1375–3020**	**$205–450**	**$205–450**	**$205–450**	**$205–450**

Figure 4: TCO for Proprietary CMS (all figures in '000)

- What are the costs associated with your content being unavailable either through the Website or your primary content storage systems?
- What's the risk of having inaccurate content on your Website?
- How much does the insurance for that risk cost?
- How do you recover and replace inaccurate content when your Webmaster is unavailable?

Although it's challenging to put a value tag on these intangibles, these cost savings relate directly to ROI. Depending on the type of organization and the size and type of the asset management and Web publishing requirements, measurable ROI can be predicted and proved. Alternatively, organizations can start to look at the TCO of implementing a CMS, both proprietary and OSS.

Several components determine the final price of a CMS implementation. Figure 4 shows costs incurred over five years when using a proprietary CMS system. Most proprietary systems run on expensive servers running a proprietary OS such as Solaris or Windows 2000. License fees for proprietary systems range from $600,000 to $2 million. Organizations seeking annual support for these products usually include an industry standard of 18.5 percent for a license fee and solution support costs of approximately $25,000.

Because of the one-size-fits-all nature of proprietary CMS systems, customization cycles are longer and expensive. When all other costs are factored in, a fully operational proprietary solution normally runs anywhere from $1.4 million to $3 million.

OSS systems can be an effective alternative. Because license fees don't apply, they offer an immediate value proposition. Usually, the cost to implement a CMS package is the same. Because of a vast developer community, most open source CMS solutions boast of several add-ons, directly translating into lowered customization fees. Several vendors exist that provide support for the open source code and are also responsible for training and support. Figure 5 shows costs incurred in implementing an OSS CMS solution.

	Year 1	Year 2	Year 3	Year 4	Year 5
Servers	25				
OS/database	none				
CMS software	none				
Product support	10	10	10	10	10
Implementation	300				
Customization	100	25	25	25	25
Training	50				
Application support	25	25	25	25	25
TOTAL	**$510**	**$60**	**$60**	**$60**	**$60**

Figure 5: TCO for an Open Source CMS (all figures in '000)

OSS solutions cost half as much as proprietary systems and offer similar features. Depending on the cost-savings potential, both solutions pay for themselves, although in different timelines. However, a commercial CMS solution "locks in" an organization to some platform, leading to the scalability saturation. Furthermore, there are no guarantees that the license fees will remain flat, unlike OSS solutions where the licensing component doesn't apply.

Conclusion

Content management is an important part of an enterprise Web strategy and it will only increase in importance as more business functions are moved to the Web. Specialized CMS services such as workflow and personalization will remain important selling points. For organizations considering a CMS solution, more and better options than ever exist. Choosing the best solution, though, requires increased due diligence.

Significant developments over the past year have encouraged organizations to adopt OSS solutions for their CMS needs. OSS solutions have helped organizations stay more competitive and realize high ROI. But for most organizations seeking OSS solutions, the price tag is irrelevant; the stability and the growing number of references in favor of OSS alternatives have influenced organizations to adopt these solutions. Organizations are seeing rapid reduction in system downtime by moving to a Linux server. The arguments against implementing an open source CMS usually encompass one primary concern: uncertainty. Product support, documentation, and user training are often subject to the whims of the community developers.

Deciding which way to go on your CMS implementation depends on many factors, but ultimately, you want the best ROI and the lowest TCO possible. The important variables are your requirements, resources, and the demands of your particular situation. Some of the high-priced CMS solutions look fantastic, but the outcome is what's important. Choose what's right for your situation. ●

Vasuki Kasturi is a program manager at CIGNEX Technologies, Inc., and heads the Data Integration and Business Intelligence practice.
e-mail: vasuki@cignex.com
Website: www.cignex.com

BY NATHANIEL PALMER

Open Source Will Build New Markets by First Breaking Them Apart

Vertically integrated market sectors face an inescapable utility curve, where innovation is suppressed as competition is locked out. The generation of computers prior to the PC (notably mainframes and mini-computers) followed a vertically integrated, inter-dependent architecture where all components were controlled by a single firm, and competitive differentiation was derived from tight coupling of the value chain. The result was a "winner-take-all" opportunity for monopolies, locking-out new firms with the inherent overhead required of developing an entirely new architecture and its components. The mainframe industry consolidated behind IBM, which ultimately would command 90 percent of the market, just as DEC dominated the mini-computer space.

Contrast this, however, with industry models based on a modular architecture, such as that of the PC. For the last two decades, innovation and wealth creation have been spread across the value chain of component providers such as Microsoft and Intel, plus component assemblers such as Dell, Compaq, and a host of others. The breadth and scope of the PC and desktop computing market have been far greater than any single, vertically integrated player could have managed. Although IBM participated in the PC market's rising tide, it controlled a much smaller share than it enjoyed in the mainframe business.

Software today reflects a market amid an inflection point, facing the same challenges and limitations seen in earlier vertically integrated technology sectors. Following the precedent established by previous technology evolutions and models, it's the shift from vertical integration to an open architecture (a move hastened by open source) that will save the software industry. Consider the recent success of Apple and its OS X platform, where open source has provided the catalyst for a new wave of innovation and specialized development. Traditionally one of the worst offenders of the "not-invented-here" mentality, Apple has lagged behind Windows/Intel in user adoption, as well as support by third-party application developers, despite what many conceded to be a superior platform.

With the release of its OS X environment, however, Apple is amid a sea of change in the design of both its products and business model. Apple's "Powered by Darwin" campaign, for example, is designed to allow developers to customize and enhance key Apple software, while also providing a forum for Apple engineers to collaborate directly with the open source community. This includes the core operating system commonly known as Darwin, which combines several core open source components, including Mach 3.0, an operating system services stack on the 4.4BSD (Berkeley Software Distribution of Unix). Open source also has allowed Apple to bring to market a number of desktop applications for the OS X platform, with a speed unprecedented in its earlier generations of software development. These applications have played a significant role in the growth Apple has recently experienced in the adoption of the Macintosh platform. They also have no doubt helped re-shape Apple's thinking away from proprietary processors by supporting Intel processors, starting in 2006.

The modular design of Linux provides a platform for many value-added services to be delivered at the hardware level (e.g., built-in services for connectivity and security) and to be able to work with most installed Linux distributions. Unlike proprietary platforms (notably Windows), Linux's modularity allows applications to run their own user environment, without having to expose the operating system. This phenomenon has already emerged within consumer electronics, where manufacturers have found success developing tightly packaged, off-the-shelf commercial products leveraging open source components. Examples include a Linksys 802.11g Wireless Router running a Linux kernel, the Nokia Smartphone using a custom Web browser based on the same open source project used to create Apple's Safari, and an application called OsiriX, developed by a radiologist, that allows doctors to collaborate on scanned images using an iPod as portable storage.

These initiatives have demonstrated how open source offers a time-to-market advantage that greatly exceeds that offered by the traditional commercial development model. The ability to leverage components as building blocks allows application developers to be re-cast as business innovators. These initiatives also provide the economic potential for leveraging open source for innovation beyond the traditional confines of IT and packaged software. The end game for OSS isn't simply to displace commercial Unix with Linux, but to unleash a vast array of new product capabilities that may otherwise evade discovery and invention with blinders of proprietary software. ●

Nathaniel Palmer is president of Transformation+Innovation—a consulting, education, and advisory firm that guides business strategy and transformation through the optimization of technology, knowledge management, and process redesign. He's the co-author of *The X-Economy* (Texere, May 2001) and has authored more than 200 studies and published articles.
e-Mail: npalmer@os30.com
Website: www.transformationandinnovation.com

Searching for Alternatives to Pricey, Complicated Open Systems Backup Products Can Be a Hassle.

LOOK NO FURTHER!

Reliable Backup/Recovery • Extensive Multi-Platform Support

Backup to Disk or Tape • Centralized Administration

Broad Based Device Support • Affordable License Fees

SAN Express LAN-Free Backup • Hot Database Agents

ALL FROM A VENDOR PROVIDING BACKUP SOFTWARE FOR OVER 33 YEARS!

If you are looking for alternatives to pricey, complicated open systems backup products than look no further. If you need a backup product that is simple to use, very cost effective, with a personalized technical support model, then consider **FDR/UPSTREAM's Family of Products**.

The UPSTREAM Reservoir extends the power of UPSTREAM so organizations can now utilize UPSTREAM either in mixed z/OS mainframe–open systems environments or entirely non-mainframe environments.

Thousands of sites have trusted INNOVATION's backup products for decades. Take the INNOVATION challenge and see how FDR/UPSTREAM can help you manage your UNIX and distributed backup issues.

Innovation Data Processing offers a **FREE** 90-day No-obligation trial to evaluate the product in your environment. To order the trial, request documentation or an UPSTREAM white paper, please don't hesitate to call us at **(973) 890-7300**, or email **sales@fdrinnovation.com**.

Visit us at: LinuxWorld 2006 • Booth #1030 • April 3-6, 2006 • Boston, MA

INNOVATION® DATA PROCESSING

CORPORATE HEADQUARTERS: 275 Paterson Ave., Little Falls, NJ 07424 • (973) 890-7300 • Fax: (973) 890-7147
E-mail: support@fdrinnovation.com • sales@fdrinnovation.com • http://www.innovationdp.fdr.com

EUROPEAN OFFICES:	FRANCE	GERMANY	NETHERLANDS	UNITED KINGDOM	NORDIC COUNTRIES
	01-49-69-94-02	089-489-0210	036-534-1660	0208-905-1266	+31-36-534-1660